UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE TECNOLOGÍA ELECTRÓNICA

PROYECTO FIN DE CARRERA

I.T.I: ELECTRÓNICA INDUSTRIAL

ESTUDIO DE UN SISTEMA DE VISION ESTEREO CONTROLADO POR HARDWARE

Autor: Miguel Mariano García Zamora

Director: Michael V. García Lorenz

Noviembre, 2012

Título: Estudio de un Sistema de Visión Estéreo Controlado por Hardware
Autor: Miguel Mariano García Zamora
Director: Michael V. García Lorenz

EL TRIBUNAL

Presidente:_____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

A todos mis seres queridos.

Resumen

Este proyecto consiste en el estudio de un sistema de visión estéreo controlador por Hardware.

El hardware que controlará el sistema será una **FPGA** de la marca Digilent, con un chip Xilinx Spartan-6 LX45.

La **cámara estéreo**, también de Digilent, estará compuesta por dos sensores que captaran dos imágenes a la vez, desplazadas entre sí lateralmente.

La **cámara estero** captara imágenes que será almacenadas y tratadas por la **FPGA**, para posteriormente enviarlas a través de un puerto HDMI hacia una pantalla.

Para el diseño de todo el sistema se utilizará el lenguaje de descripción hardware **VHDL** (Very High Description Language).

Para el estudio se partirá de un diseño básico desarrollado por el fabricante.

Un vez estudiado y entendido el sistema, se procederá al desarrollo de diseños relacionados con la visión estéreo y que utilicen nuestro sistema, como son el cálculo de mapas de **disparidad** y la generación de **anáglifos**.

Palabras clave: FPGA, Cámara Estero, VHDL, Disparidad, Anáglifo.

Abstract

This Project consist in the study of a stereo vision system controller by hardware.

The hardware, that will handle the system, will be a **FPGA** of Digilent, with a chip Xilinx Spartan-6 LX45.

The **stero cam**, also of Digilent, will be composed of two sensors that capture both images simultaneously, offset from each other laterally.

The **stereo cam** would capture images that will be stored and processed by the FPGA, for later sending through an HDMI port to a display.

For the design of the all system will use the hardware description language VHDL (Very High Description Language).

For the study will be based on a basic design developed by Digilent.

A once studied and understood the system, we proceed to the development of related designs stereo vision and that using our system, like calculating **disparity** maps and generation of **anaglyphs**.

Keyword: FPGA, Stereo Cam, VHDL, Disparity, Anaglyphs.

Índice general

Capítulo Página

Capítulo Página

Capítulo Página

Índice de figuras

Índice de tablas

1. INTRODUCCION Y OBJETIVOS

1.1 INTRODUCCION

Este proyecto titulado *"Estudio de un Sistema de Visión Estéreo Controlado por Hardware"* se centra en el desarrollo de un sistema de captación y tratamiento de imágenes controlado exclusivamente por un Hardware.

El diseño de este sistema viene motivado por el hecho de intentar realizar un dispositivo de captación y tratamiento de imágenes más rápido que los existentes, generalmente controlados por software, y con una velocidad de ejecución más lenta que la de nuestro sistema.

Para el control por hardware, se ha elegido una FPGA, de la marca Digilent, con un chip Xilinx Spartan-6 LX45, que será programada mediante el lenguaje de descripción hardware VHDL

El dispositivo de captación es una cámara estéreo, también de la marca Digilent, compuesta por dos sensores CMOS Aptina MT9D112, de iguales características y configuración, desplazados entre sí horizontalmente.

Indicar que para el desarrollo de nuestro sistema, se ha tomado como referencia un diseño básico, de ejemplo, creado por el fabricante.

1.2 OBJETIVOS

El objetivo fundamental de este proyecto es el de desarrollar un sistema de visión estéreo controlado por hardware; el objetivo es estudiar, diseñar y desarrollar un sistema que, al contrario que los habituales sistemas de visión estéreo controlados por software, esté controlado por hardware, con lo que se reduzca considerablemente el tiempo de ejecución del proceso de captura y tratamiento de imágenes.

En base a este objetivo principal, se proponen los siguientes objetivos parciales:

- Estudiar y comprender el diseño básico de ejemplo creado por el fabricante.

Será necesario estudiar y comprender cada uno de los módulos que componen el diseño básico, así como estudiar los resultados; se deberá entender como se configuran los sensores de la cámara, cómo y cuando se capturan las imágenes, como se almacenan, como se muestran por pantalla, etc.

- Estudiar y desarrollar algunos posibles usos para nuestro Sistema de Visión Estero Controlado por Hardware.

Como objetivos secundarios, se intentará usar nuestro sistema de visión para desarrollar diversos diseños relacionados con la visión estéreo. Se han propuesto dos usos: un diseño que genere mapas de disparidad, imágenes en las que el nivel de gris es proporcional a la disparidad e inversamente proporcional a la profundidad; y un segundo diseño que genere anáglifos, imágenes de dos dimensiones capaces de provocar un efecto tridimensional, cuando se ven con lentes especiales.

A continuación puede verse un diagrama de bloques de nuestro sistema:

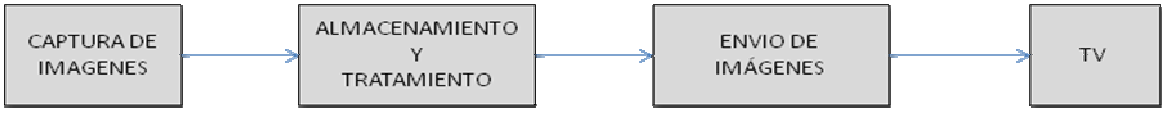


Figura 1: Diagrama de Bloques

Los componentes Hardware necesarios para la implementación de lo descrito anteriormente son:

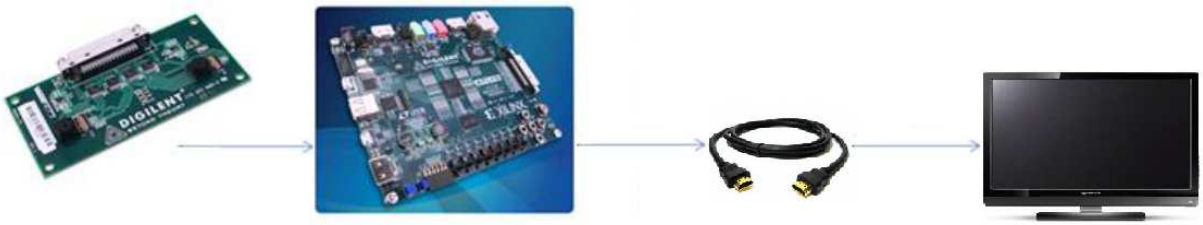


Figura 2: Componentes Hardware

1.3 ESTRUCTURA DE LA MEMORIA

Para facilitar la lectura de la memoria, a continuación se incluye un resumen de cada capítulo.

Estado del Arte: En este apartado se presentaran los fundamentos teóricos de las FPGAs, del lenguaje VHDL así como de la visión estéreo, de los Mapas de disparidad y de los Anáglifos, con el fin de establecer una base teórica sobre realizado en este proyecto.

Descripción previa del Sistema: En este apartado se hará una descripción detallada de los elementos específicos con los que se ha realizado el proyecto: *FPGA Spartan-6* y cámara estéreo *VmodCAM*, ambas del fabricante Digilent. Además, se describirá el programa *Xilinx ISE Design Suite*, que ha sido el programa con el que se ha programado la FPGA.

Arquitectura de Diseño: En este apartado nos centraremos en el verdadero desarrollo del proyecto. Se empezará describiendo cada uno de los módulos de los que consta el programa de partida, proporcionado por el fabricante**,** el cual ha sido la base de nuestro proyecto. A continuación, se describirán los algoritmos de Mapa de Disparidad y de generación de Anáglifos. Y finalmente, se expondrá, como ha sido realizada la implementación de todos los diseños finales.

Resultados: En este apartado se comentaran y presentaran todos los resultados obtenidos ene le proyecto, añadiendo la presentación y explicación de imágenes, que ayuden a comprender todos los resultados.

Conclusiones: En este apartado se expondrán las conclusiones sobre el proyecto y sus resultados, los desafíos que ha supuesto, así como posibles trabajos futuros que continúen con la labor comenzado con este proyecto.

Bibliografía: Aquí se listarán las referencias de los artículos, libros y webs consultadas para la realización de este proyecto.

2. ESTADO DEL ARTE

2.1 INTRODUCCION

En este apartado se presentaran los fundamentos teóricos de las FPGAs, del lenguaje VHDL, así como de la visión estéreo, de los Mapas de disparidad y de los Anáglifos, con el fin de establecer una base teórica sobre realizado en este proyecto.

2.2 PLACA FPGA

Una FPGA (*Field Programmable Gate Array*) es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada *'in situ'* mediante un lenguaje de descripción especializado. La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinacional hasta complejos sistemas en un chip.

Las FPGAs tienen la ventaja de ser reprogramables, tener unos bajos costes de desarrollo y adquisición y un tiempo de desarrollo bastante menor, en comparación a otros dispositivos de usos similares.

2.2.1 Historia

Las FPGAs fueron inventadas en el año 1984 por Ross Freeman y Bernard Vonderschmitt, co-fundadores de Xilinx, y surgen como una evolución de los CPLDs (Complex Programmable Logic Device).

Tanto los CPLDs como las FPGAs contienen un gran número de elementos lógicos programables. Si medimos la densidad de los elementos lógicos programables en puertas lógicas equivalentes podríamos decir que en un CPLD hallaríamos del orden de decenas de miles de puertas lógicas equivalentes y en una FPGA del orden de cientos de miles hasta millones de ellas.

Aparte de las diferencias en densidad entre ambos tipos de dispositivos, la diferencia fundamental entre las FPGAs y los CPLDs es su arquitectura. La arquitectura de los CPLDs es más rígida y consiste en una o más sumas de productos programables cuyos resultados van a parar a un número reducido de biestables síncronos. La arquitectura de las FPGAs, por otro lado, se basa en un gran número de pequeños bloques utilizados para reproducir sencillas operaciones lógicas, que cuentan a su vez con biestables síncronos. La enorme libertad disponible en la interconexión de dichos bloques confiere a las FPGAs una gran flexibilidad.

Otra diferencia importante entre FPGAs y CPLDs es que en la mayoría de las FPGAs se pueden encontrar funciones de alto nivel embebidas en la propia matriz de interconexiones, así como bloques de memoria.

2.2.2 Características

Una jerarquía de interconexiones programables permite a los bloques lógicos de un FPGA ser interconectados según la necesidad del diseñador del sistema. Estos bloques lógicos e

interconexiones pueden ser programados después del proceso de manufactura por el usuario/diseñador, así que el FPGA puede desempeñar cualquier función lógica necesaria.

Una tendencia reciente ha sido combinar los bloques lógicos e interconexiones de los FPGA con microprocesadores y periféricos relacionados para formar un «Sistema programable en un chip». Ejemplo de tales tecnologías híbridas pueden ser encontradas en los dispositivos Virtex-II PRO y Virtex-4 de Xilinx, los cuales incluyen uno o más procesadores PowerPC embebidos junto con la lógica del FPGA. El FPSLIC de Atmel es otro dispositivo similar, el cual usa un procesador AVR en combinación con la arquitectura lógica programable de Atmel. Otra alternativa es hacer uso de núcleos de procesadores implementados haciendo uso de la lógica del FPGA. Esos núcleos incluyen los procesadores MicroBlaze y PicoBlaze de Xlinx, Nios y Nios II de Altera, y los procesadores de código abierto LatticeMicro32 y LatticeMicro8.

Muchas FPGA modernas soportan la reconfiguración parcial del sistema, permitiendo que una parte del diseño sea reprogramada, mientras las demás partes siguen funcionando. Este es el principio de la idea de la «computación reconfigurable», o los «sistemas reconfigurables».

2.2.3 Programación

La tarea del programador es definir la función lógica que realizará cada uno de los CLB, seleccionar el modo de trabajo de cada IOB e interconectarlos.

El diseñador cuenta con la ayuda de entornos de desarrollo especializados en el diseño de sistemas a implementarse en un FPGA. Un diseño puede ser capturado ya sea como esquemático, o haciendo uso de un lenguaje de programación especial. Estos lenguajes de programación especiales son conocidos como HDL o *Hardware Description Language* (lenguajes de descripción de hardware). Los HDLs más utilizados son: VHDL (que será el que utilicemos), Verilog y ABEL.

2.2.4 Aplicaciones

Cualquier circuito de aplicación específica puede ser implementado en un FPGA, siempre y cuando esta disponga de los recursos necesarios. Las aplicaciones donde más comúnmente se utilizan los FPGA incluyen a los DSP (procesamiento digital de señales), radio definido por software, sistemas aeroespaciales y de defensa, prototipos de ASICs, sistemas de imágenes para medicina, sistemas de visión para computadoras (nuestro objetivo), reconocimiento de voz, bioinformática, emulación de hardware de computadora, entre otras. Cabe destacar que su uso en otras áreas es cada vez mayor, sobre todo en aquellas aplicaciones que requieren un alto grado de paralelismo.

2.3 LENGUAJE VHDL

Con el objeto de permitir una gran flexibilidad en el diseño de sistemas electrónicos se desarrolló el lenguaje de descripción hardware VHDL (*Very High Speed Integrated Circuit (VHSIC) Hardware Description Languaje),* siendo aprobado por el *IEEE (Institute of Electrical and Electronics Engineeres, Inc.*) en 1987 [IEE87] y revisado en 1993[IEE93]. Este

lenguaje ofrece la facilidad de desarrollo de los lenguajes de alto nivel al mismo tiempo que potencia en el manejo del hardware que se programa con él. El VHDL fue desarrollado como un lenguaje para el modelado y simulación lógica dirigida por eventos de sistemas digitales, y actualmente se utiliza también para la síntesis automática de circuitos.

El VHDL es un lenguaje con una sintaxis amplia y flexible que permite el modelado estructural, en flujo de datos y de comportamiento hardware. También permite el modelado preciso del comportamiento de un sistema digital conocido y el desarrollo de modelos de simulación.

Algunas ventajas del uso de VHDL para la descripción hardware son:

1. Permite diseñar, modelar y comprobar un sistema desde un alto nivel de abstracción bajando hasta el nivel de definición estructural de puertas

2. Los circuitos descritos utilizando VHDL, siguiendo unas guías para síntesis determinadas, pueden ser utilizados por diversas herramientas de síntesis para crear e implementar circuitos.

3. Los módulos creados en VHDL pueden utilizarse en diferentes diseños, lo que permite la reutilización del código.

4. VHDL permite un diseño Top-down, esto es, describir el comportamiento de los bloques en alto nivel, analizarlos y refinar la funcionalidad en alto nivel, antes de llegar a niveles más bajos de abstracción de la implementación del diseño.

2.4 VISION ESTEREO

2.4.1 La estereoscopia

La estereoscopía es cualquier técnica para conseguir información tridimensional de una imagen creando una ilusión de profundidad de la misma. Para ello se captan dos o más imágenes en 2D.

La manera más sencilla de obtener una ilusión tridimensional consiste en captar dos imágenes con diferentes ángulos de captación, imitando al sistema visual del ser humano, el cual el ojo izquierdo capta una imagen ligeramente diferente al del ojo derecho. Esto se puede comprobar mirando a un objeto fijamente y tapándose, primero el ojo izquierdo y después el derecho, y observando que la perspectiva del objeto cambia. El cerebro es el encargado de juntar las dos imágenes para obtener así una sensación espacial. Con este ejercicio de taparse el ojo izquierdo y luego el derecho, se observa también, que la imagen sigue manteniendo su ilusión de profundidad. Esto se debe a que el cerebro realiza una construcción intuitiva de la imagen gracias a varios factores que se verán a continuación. Este tipo de ilusión tridimensional que se consigue con una sola imagen, se le denomina visión monocular.

Figura 3: Imágenes estéreo. Imágenes izquierda y derecha de la misma escena con un desplazamiento horizontal (hacia la derecha) de la cámara. [3]

Los factores que consiguen que el cerebro asimile la imagen 2D como una imagen 3D, son los siguientes:

- Distribución de luces y sombras sobre el objeto: La iluminación es un factor intuitivo del volumen muy importante, ya que la sombra y el contraste nos aportan gran sensación de relieve.

- Superposición de imágenes: Cuando un objeto se encuentra en superposición a otro, es decir, se encuentra en la realidad ante otro, el más cercano (delante) cubre al más lejano (detrás).

- Perspectiva: El efecto de *perspectiva* produce una clara sensación de profundidad. Las líneas paralelas horizontales parecen converger en el horizonte.

- Diplopía fisiológica: Para que el cerebro pueda interpretar una imagen en tercera dimensión, requiere de datos sobre la distancia de los objetos. Dicha información se obtiene gracias a que tenemos dos ojos, así cada uno de ellos percibe los elementos de la escena desde un ángulo distinto, dando como resultado una triangulación de la cual el cerebro obtiene la distancia al objeto.

- Movimiento de paralelaje: El desplazamiento del observador produce la impresión de que se mueven los objetos de la escena en un sentido u otro dependiendo de su posición.

Algunas de las técnicas para conseguir una percepción de imagen tridimensional a partir de imágenes 2D, son las siguientes:

- Anáglifos: Los anáglifos son estereofotografías tomadas o tratadas con filtros de distintos colores sobrepuestas en una sola imagen. Se observan por medio de unas gafas llamadas gafas anáglifo y que tienen un filtro de diferente color para cada ojo. La misión de estos filtros es hacer llegar a cada ojo únicamente la imagen que le corresponde.

Figura 4: Gafas anáglifo

- Polarización: Se utiliza luz polarizada para separar las imágenes izquierda y derecha. El sistema de polarización no altera los colores, aunque hay una cierta pérdida de luminosidad. Se usa tanto en proyección de cine 3D como en monitores de ordenador mediante pantallas de polarización alternativa. Hoy día es el sistema más económico para una calidad de imagen aceptable.

- Sistema Cromatek: El sistema cromatek utiliza lo que se conoce como rejilla de difracción. La rejilla de difracción funciona de manera semejante a un prisma de cristal: la luz que la atraviesa se descompone en colores que cambia de angulación según su tonalidad ya que ésta, está asociada a su frecuencia y por tanto, a su longitud de onda.

  Éste cambio de ángulo que cada color sufre al ser difractado incide en el ojo y hace que los objetos parezcan tener una profundidad distinta según su color. El inconveniente es que para que la desviación del ángulo al difractarse sea notoria respecto a la luz directa que llega al otro ojo, las imágenes tienen que tener colores intensos; por lo que el rango cromático que podremos utilizar queda limitado. [4]

- Efecto Pulfritch: El sistema pulfritch está basado en un dato fisiológico respecto al cerebro y dice que éste tarda un poco más en procesar las imágenes oscuras que las claras. Así si se pone un filtro oscuro en un solo ojo y se observa un objeto en movimiento, el cerebro tardará más tiempo en procesar las imágenes procedentes de este ojo. Por lo que si la escena que se observa está en continuo movimiento lateral, la imagen del ojo con filtro parecerá estar en una posición o ángulo distinto con respecto al observado directamente sin filtro, que tendrá la imagen procesada instantes antes.

Todos los métodos anteriores, disponen las imágenes para que el cerebro haga la función de juntarlas y así obtener la imagen 3D, pero si se quiere obtener la transformación de la imagen en 3D sin ayuda del cerebro se tendrá que recurrir a modelos matemáticos y a algoritmos que permitan construir la imagen en 3D, a partir de las imágenes en 2D.

Uno de los algoritmos más sencillos para la obtención de las imágenes en 3D se basa en el cálculo de la profundidad, a partir de la información obtenida de las imágenes en 2D.

El proceso de obtención de la imagen en 3D es obtener dos imágenes en 2D, identificar los puntos correspondientes en ambas imágenes. Posteriormente mediante geometría se Calcula la profundidad (Z) de cada punto – en base a la distancia entre los puntos

Correspondientes (disparidad) en las imágenes y los parámetros de los objetivos de las Cámaras (longitud focal, distancia entre objetivos).

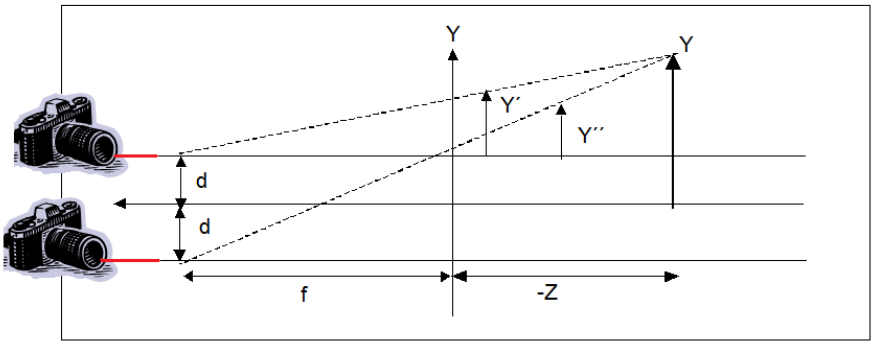


Figura 5: Geometría del cálculo de profundidad

Donde la fórmula para obtener la profundidad Z es:

$$Z = f - \frac{2df}{(y'-y'')}$$

Como se puede observar el algoritmo es sencillo, lo realmente complejo en el proceso de construcción de una imagen en 3D es la de identificar la correspondencia entre puntos.
Dos de las maneras de encontrar la correspondencia de puntos son:

- Estéreo "denso" – se hace la correspondencia en toda la imagen, a nivel pixel (template matching, relajación,…).

- Basado en características – se busca la correspondencia entre ciertas características o patrones distintivos (orillas, esquinas, SIFT,…).

La primera de las maneras utiliza algoritmos bastante más complejos, con un esfuerzo computacional importante, mientras que el segundo realiza los mismos algoritmos que el método anterior, pero sólo se aplica a puntos importantes, como bordes, esquinas, etc.
Este método no realiza tanto esfuerzo computacional pero la calidad de la imagen se reduce bastante.

2.5 LA DISPARIDAD

Una forma de estimar la profundidad de cada uno de los puntos en la escena es mediante el cálculo de la disparidad entre las imágenes de la misma. Asumiremos que la escena es estática, es decir, los objetos visibles en la escena no cambien su posición en la misma ni sufren deformaciones.

Para definir la disparidad asumamos una configuración de dos cámaras de características similares, como la que se muestra en la figura 6. Estas dos cámaras forman un par estéreo, y asumiremos que cada una de ellas cumplen un modelo pinhole. Los ejes ópticos de las

cámaras son paralelos, $\overrightarrow{O_R o_R} \parallel \overrightarrow{O_L o_L}$. Ambas cámaras tienen la misma distancia focal, f, con centros OL y OR separados una distancia B, llamada línea base (baseline), de forma que las imágenes que se forman, IL e IR, estén en planos paralelos. De esta manera la línea base es paralela a la coordenada x de las imágenes. Con el modelo pinhole considerado, un punto en el espacio tridimensional P, con coordenadas $(X, Y, Z)^T$, se proyecta en cada una de las imágenes bidimensionales en los puntos pL y pR, con coordenadas $(xL, yL)^T$ y $(xR, yR)^T$, respectivamente.
El plano que contiene a los puntos P, OL y OR, interseca a las imágenes en dos rectas eL y eR. Dada la configuración del par estéreo, estas son rectas epipolares entre sí, o sea, un punto, pL, en la recta eL de la imagen IL tiene su correspondiente en algún punto de la recta eR. Esto reduce la búsqueda del correspondiente de pL de toda la imagen IR a la recta eR.

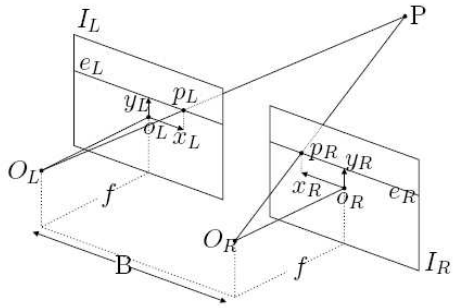


Figura 6: Configuración de las cámaras del par estéreo.

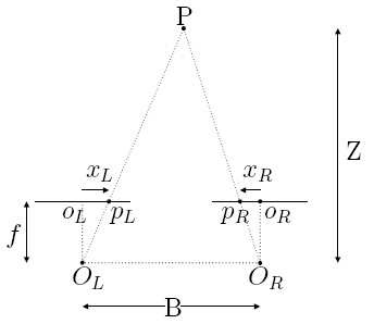


Figura 7: Relación geométrica entre los parámetros del par estéreo para obtener la profundidad Z a partir de la disparidad d.

En la figura 7 podemos ver como se relacionan los parámetros definidos en el par estéreo, que permiten obtener la relación entre la disparidad d y la profundidad Z del punto P.
La disparidad es la diferencia en las coordenadas horizontales de los puntos pL y pR, o sea, d = xL - xR. Dependiendo el sistema de referencia utilizado en las imágenes, la definición puede cambiar de forma que el signo sea siempre positivo. Las coordenadas de pL y pR quedan relacionadas mediante:

$$\begin{cases} x_L = x_R + d \\ y_L = y_R \end{cases}$$

Considerando los triángulos POLOR, pRoROR y pLoLOL, utilizando semejanza entre triángulos se llega a:

$$d = \frac{f}{Z} B$$

Entonces, se tiene la relación entre d y Z:

$$d \propto \frac{1}{Z}$$

Basados en la ecuación anterior podemos recuperar, a menos de una constante de escala, la profundidad de cada pixel en cada una de las imágenes a partir de la disparidad calculada.

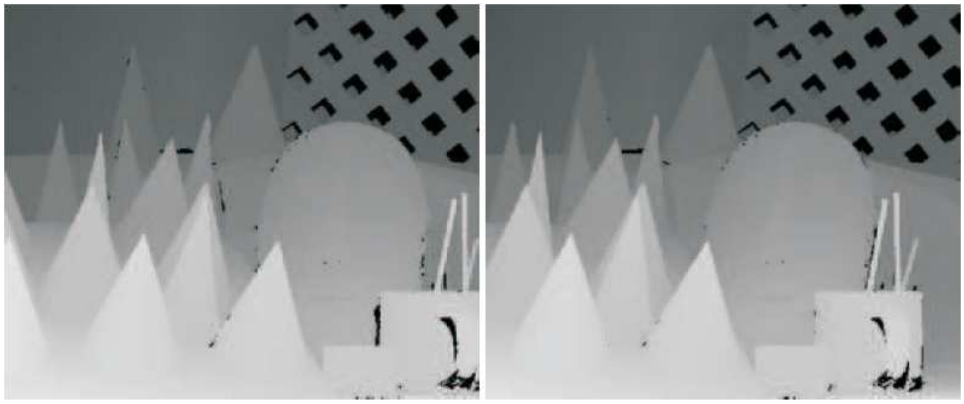


Figura 8: Mapa de disparidad reales para las imágenes de la figura 3; el nivel de gris es proporcional a la disparidad e inversamente proporcional a la profundidad. En los puntos negros la disparidad es desconocida. [3]

La relación de proporcionalidad inversa planteada en la ecuación anterior es fácilmente verificable observando alternadamente las imágenes izquierda y derecha, y notando que los objetos más cercanos a la cámara -menor Z- tienen mayor desplazamiento relativo – mayor d- en las imágenes (ver figura 3).

Los algoritmos de cálculo de disparidad obtienen una imagen con el valor de disparidad calculado en cada punto de las imágenes de entrada. Estas imágenes se conocen como Mapas de Disparidad; en la figura 8 se ven los Mapas de Disparidad reales de las imágenes de la figura 3.

2.6 ANAGLIFO

Las imágenes de anáglifo o anáglifos son imágenes de dos dimensiones capaces de provocar un efecto tridimensional, cuando se ven con lentes especiales.

Se basan en el fenómeno de síntesis de la visión binocular y fue patentado por Louis Ducos du Hauron en el 1891 con el nombre de este artículo. Las imágenes de anáglifo se

componen de dos capas de color, superpuestas pero movidas ligeramente una respecto a la otra para producir el efecto de profundidad. Usualmente, el objeto principal está en el centro, mientras que lo de alrededor y el fondo está movidos lateralmente en direcciones opuestas. La imagen contiene dos imágenes filtradas por color, una para cada ojo. Cuando se ve a través de las Gafas anáglifo, se revelará una imagen tridimensional. La corteza visual del cerebro fusiona ambas imágenes, haciendo que percibamos una escena con profundidad.

3. DESCRIPCION PREVIA DEL SISTEMA

3.1 INTRODUCCION

En este apartado se describe cada uno de los elementos específicos con los que se ha realizado el proyecto: la *FPGA Spartan-6* [1] y la cámara estéreo *VmodCAM* [2], del fabricante Digilent y el entorno de trabajo, el programa *Xilinx ISE Design Suite*.

3.2 FPGA

3.2.1 Introducción

La FPGA que se ha utilizado es la FPGA Xilinx Spartan-6 LX45, la cual viene integrada en la placa Atlys. La placa Atlys es un completo sistema preparado para el desarrollo de circuitos digitales. Tiene un gran número de periféricos entre los que destacan Gbit Ethernet, HDMI Video, 128MByte 16-bit de memoria DDR2, y puertos USB y de audio.

La placa Atlys es compatible con todas las herramientas de Xilinx CAD, incluyendo ChipScope, EDK, y el software libre ISE WebPack ™.

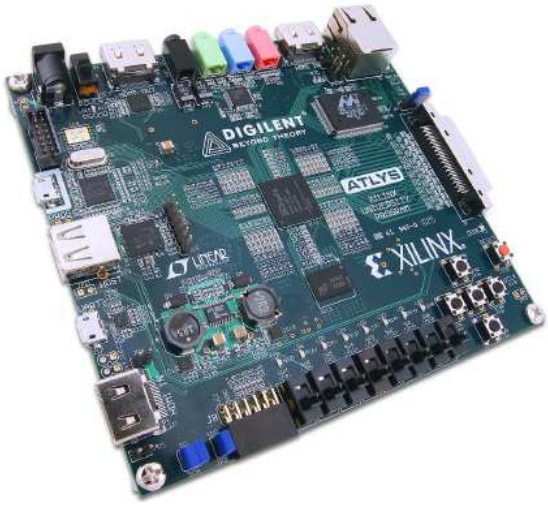


Figura 9: Placa Atlys

El Chip LX45 Spartan-6 está optimizado para la lógica de alto rendimiento y ofrece:

- 6.822 segmentos, cada uno con cuatro LUT de 6 entradas y ocho flip-flops
- 2.1Mbits de RAM
- Cuatro relojes cerámicos (ocho DCMs y cuatro PLLs)
- Seis bucles de enganche de fase
- Mas de 500MHz de velocidad de reloj

La placa Atlys incluye un nuevo sistema Adept Digilent de USB2, que ofrece al dispositivo programación, monitoreo de alimentación y consumo en tiempo real, pruebas automatizadas de la placa y simulación de entradas /salidas.

Características

• Paquete Xilinx Spartan-6 LX45 FPGA, 324-pin BGA

• Memoria de 128MByte DDR2 de 16-bit.

• Conexión Ethernet PHY 10/100/1000

• Puertos USB2 para la programación y la transferencia de datos de programación

• Puertos USB-UART y USB-HID (para ratón / teclado)

• Dos puertos de entrada de vídeo HDMI y un puerto de salida HDMI

• Códec AC-97 con entrada y salida de línea, entrada de micrófono y salida de auriculares

• Monitores en tiempo real la alimentación.

• Memoria de 16 MByte x 4 SPI Flash para el almacenamiento de datos y de configuración

• Oscilador CMOS de 100MHz

• 48 entradas / salidas enrutadas a los conectores de expansión

• Control GPIO que incluye ocho LEDs, seis botones y ocho interruptores deslizantes

• Fuente de alimentación de 20W y cable USB

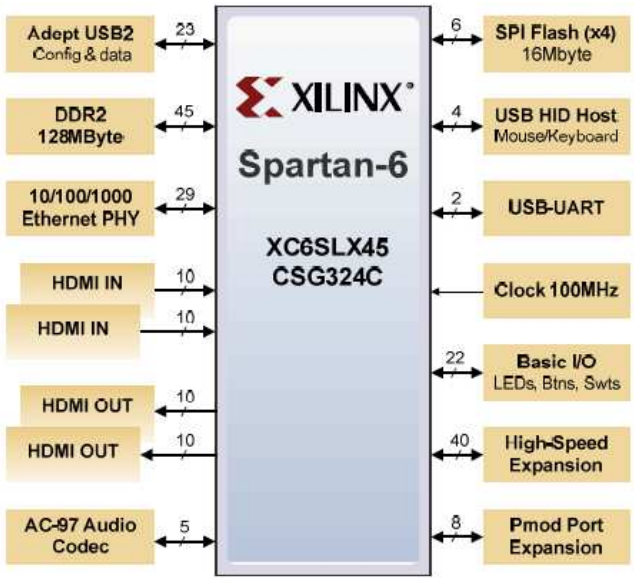


Figura 10: Esquema de Periféricos en la placa Atlys

3.2.2 Configuración

Tras el encendido, la FPGA de la placa Atlys se debe configurar (o programar) antes de que pueda realizar cualquier función. La FPGA se puede configurar de tres maneras:

- Desde el PC, mediante USB, se puede configurar la placa a través del puerto JTAG en cualquier momento, siempre y cuando este la placa encendida.
- Mediante un archivo de configuración almacenado en la SPI Flash ROM, el cual puede ser transferido automáticamente a la FPGA al encenderse.
- Mediante un pendrive de memoria USB conectado al puerto USB HID también se puede transferir un archivo a programar.

Un puente en el jumper JP11, selecciona entre JTAG o USB y memoria ROM como modos de programación. Si JP11 no está conectado, la FPGA se configura automáticamente cogiendo el programa que este guardado en la memoria ROM, en cambio, si JP11 está conectado, la FPGA permanecerá inactiva tras el encendido a la espera de que se le configure un programa desde el puerto de programación JTAG o el Puerto USB HID.

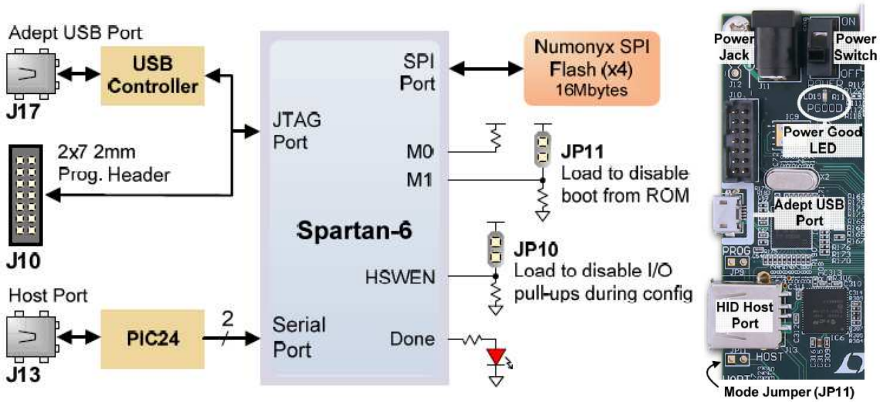


Figura 11: Esquema de Configuración de la Placa Atlys

Mantenga siempre el jumper JP12 conectado (ya sea con 3,3 V o 2,5 V). Si el jumper JP12 no está conectado, el banco 2 de la FPGA no estará alimentado, como tampoco lo estarán los pull-ups para CCLK, DONE, PROGRAM_B e INIT_B.

Los archivos de programación se almacenan dentro de la FPGA en las células de memoria SRAMs. Estos datos definen las funciones lógicas de la FPGA y las conexiones del circuito y siguen siendo válidas hasta que se borran al desconectar la alimentación o se hace valer la entrada PROG_B, o hasta que sea reemplazado por un nuevo archivo de configuración.

Los archivos de configuración que son transferidos a la FPGA a través del puerto JTAG utilizan la extensión *.bin o *.sfv; los archivos transferidos desde una memoria USB utilizan el tipo de archivo *.bit y los archivos para programar través de la memoria SPI ROM utilizan la extensión *.bit, *.bin o *.mcs. El programa ISE de Xilinx WebPack y el software EDK pueden crear archivos con las extensión *.bit, *.bin o *.mcs desde archivos VHDL, Verilog, o esquemáticos. El software Adept Digilent y el software de Xilinx iMPACT se pueden utilizar para programar la FPGA o la ROM utilizando el puerto USB Adept.

Durante la programación de la FPGA, un archivo *.bit o *.sfv se transfiere desde el PC directamente a la FPGA a través del puerto USB-JTAG. Cuando la programación, es a través de la memoria ROM, un archivo *.bit, *.bin o *.mcs se transfiere a la ROM en un proceso que consta de dos pasos. En primer lugar, la FPGA se programa con un circuito que pueda programar la memoria SPI ROM, y luego se transfieren los datos a la memoria ROM a través del circuito FPGA (esta complejidad está oculta y solo se muestra una simple interfaz). Después de que la memoria ROM ha sido programada, automáticamente se puede configurar la FPGA en un posterior encendido o reiniciar el evento si el jumper JP11 está desconectado. Un archivo de programación almacenado en la memoria SPI ROM se mantendrá hasta que se sobrescriba, independientemente de la alimentación

La FPGA se puede programar desde un lápiz de memoria conectado al puerto USB-HID si el stick contiene un solo archivo de configuración *.bit en el directorio raíz, si JP11 está conectado y si la placa está encendida. La FPGA rechazará automáticamente cualquier archivo *.bit que no se haya hecho expresamente para esta FPGA.

3.2.3 Fuente de Alimentación

La placa Atlys requiere alimentación de 5v y 4A o superior mediante un cable coaxial de centro-positivo de 2,1 mm de diámetro interno. Los circuitos de tensión del regulador de Linear Technology crean los 3.3V, 2.5V, 1.8V, 1.0V, 0.9V desde de la fuente de 5V principal. La siguiente tabla proporciona información adicional (las corrientes típicas dependen en gran medida de la configuración de la FPGA y los valores proporcionados son típicos para un diseño de tamaño/velocidad media.

Atlys Power Supplies			
Supply	**Circuits**	**Device**	**Amps (max/typ)**
3.3V	FPGA I/O, video, USB ports, clocks, ROM, audio	IC16: LT3501	3A / 900mA
2.5V	FPGA aux, VHDC, Ethernet PHY I/O, GPIO	IC15: LTC3546	1A / 400mA
1.2V	FPGA core, Ethernet PHY core	IC15: LTC3546	3A / 0.8 – 1.8A
1.8V	DDR & FPGA DDR I/O	IC16: LT3501	3A / 0.5 -- 1.2A
0.9V	DDR termination voltage (V_{TT})	IC14: LTC3413	3A / 900mA

Tabla 1: Alimentación de la Placa Atlys

Las cuatro líneas principales de tensión en la placa Atlys usan Linear Technology LTC2481 Delta-Sigma 16-bit ADC para medir continuamente el suministro de corriente. Con una precisión del 1%, estos valores medidos se pueden visualizar en un PC usando el medidor de potencia que es una parte del software Adept.

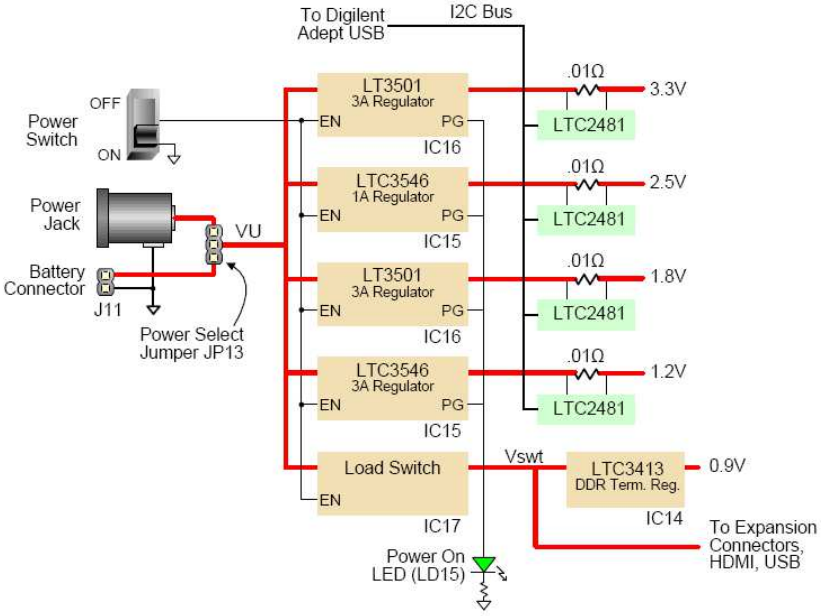


Figura 12: Esquema de Alimentación de la Placa Atlys

Los suministros de energía de la placa Atlys están habilitados por el interruptor de nivel lógico (SW8). Un LED de buen funcionamiento (LD15), indica que todas las alimentaciones están operando dentro del 10% del valor nominal.

Un interruptor de carga (en el FDC6330 IC17) pasa el voltaje de entrada VU al nodo Vswt cada vez que el interruptor de alimentación (SW8) está habilitado. Vswt se supone que es de 5V, y es utilizado por muchos sistemas en la placa, incluyendo los puertos HDMI, bus I^2C y USB host. Vswt también está disponible en los conectores de expansión, por lo que cualquier tarjeta conectada puede desactivarse junto con la placa Atlys.

3.2.4 Memoria DDR2

Un solo chip de memoria DDR2 de 1 Gb es accionada desde el bloque controlador de memoria en la FPGA Spartan-6. El dispositivo DDR2, un Micron MT47H64M16-25E o equivalente, ofrece un bus de 16 bits y 64M de aréas. La placa Atlys puede llegar a funcionar usando la memoria DDR2 con una tasa de datos de hasta 800MHz.

La interfaz de memoria DDR2 sigue el pinout y las directrices de enrutamiento especificados en la Guía del usuario del Generador de Xilinx Interfaz de memoria (MIG). La interfaz es compatible con la señalización SSTL18 y todas las direcciones, datos, relojes, y señales de control concuerdan en retraso e impedancia. Las direcciones y señales de control terminan a través de resistencias de 47-ohm conectadas a un VTT de 0.9V, y las señales de datos utilizan la función On-Die-Termination (ODT) del chip DDR2.

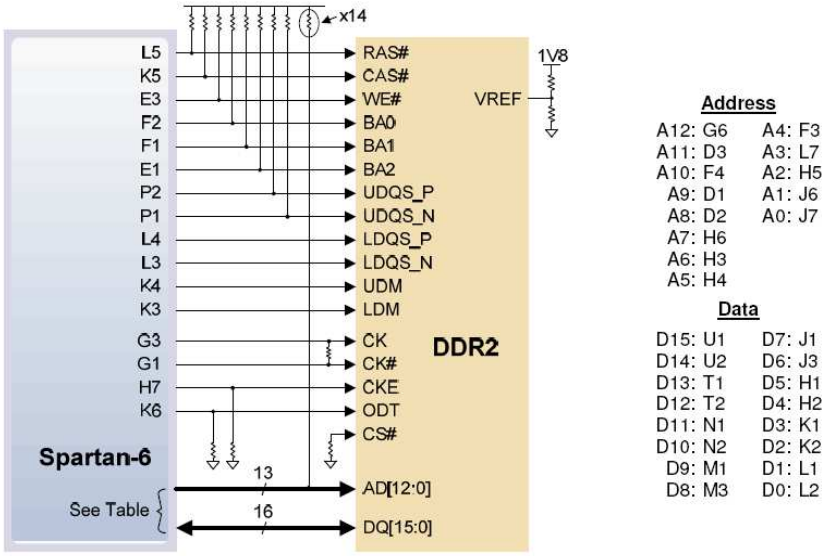


Figura 13: Esquema de de la memoria DDR2 de la placa Atlys

La placa Atlys utiliza una memoria Flash Numonyx N25Q12 Serial de 128Mbit (organizado como 16-bit por 16Mbytes) para almacenamiento no volátil de los archivos de configuración de la FPGA. El SPI Flash puede ser programada con archivos *.bit, *.bin. ó *.mcs utilizando el software Adept. Un archivo de configuración de la FPGA requiere menos de 12Mbits, dejando 116Mbits disponible para datos de usuario. Los datos pueden ser transferidos desde un PC hacia ó desde la aplicación de usuario Flash, o por instalaciones creadas en el software Adept. Los diseños programados por el usuario en la FPGA también pueden transferir datos hacia y desde la ROM.

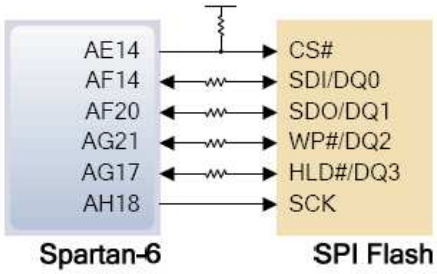


Figura 14: Esquema de la memoria Flash en la placa Atlys

3.2.5 Puerto Ethernet PHY

La placa incluye un Atlys Marvell Alaska Tri-mode PHY (el 88E1111) combinado con un halo HFJ11-1G01E RJ-45. Ambos modos de interfaz, MII como GMII, son compatibles a 10/100/1000 Mb/s. Los ajustes predeterminados utilizados en el encendido o reinicio son:

- Modo de interfaz de cobre MII / GMII
- Negociación automática activada, la publicación de todas las velocidades, prefiriendo Esclavo

- Selección interfaz MDIO, dirección PHY MDIO = 00111
- Sin pausa asimétrica, sin pausa MAC, cruce automático activado
- Detección de energía en el cable desactivado (modo de espera desactivado), polaridad de interrupción LOW

Los diseños basados en EDK pueden acceder a la PHY utilizando ya sea el diseño con núcleo IP xps_ethernetlite de 10/100 Mbps, ó diseños con el núcleo IP xps_ll_temac 10/100/1000 Mbps. El núcleo IP xps_ll_temac utiliza el núcleo duro Ethernet MAC de hardware incluido en el Virtex-5 FPGA.

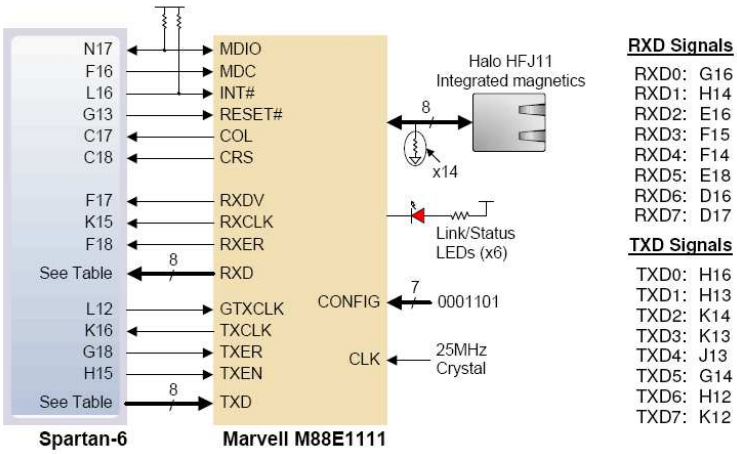
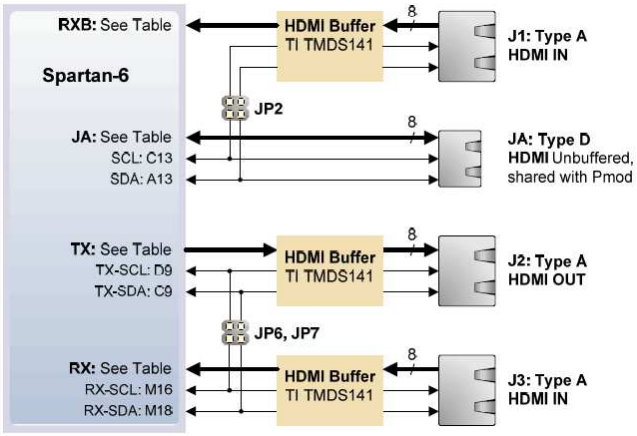


Figura 15: Esquema de conexión Ethernet en placa Atlys

3.2.6 Entrada y Salida de video (Puertos HDMI)

La placa Atlys contiene cuatro puertos HDMI, dos HDMI de entrada/salida con buffer, un puerto de salida HDMI también con buffer y un puerto sin búfer que puede ser de entrada o de salida. Los tres puertos HDMI con búfer utilizan conectores tipo A y el puerto sin buffer utiliza un conector de tipo D cargado en el lado inferior de la PCB justo debajo de los conectores PMod (tener en cuenta que el conector de tipo D es mucho menor que el tipo A). Las señales de datos en el puerto sin memoria intermedia se comparten con un conector PMod.

Los conectores HDMI de 19-pines incluyen cuatro canales de datos diferenciales, cinco conexiones GND, un solo cable de bus Consumer Electronics Control (CEC), uno de dos hilos Display Data Channel (DDC) de bus, que es esencialmente un bus I^2C, un Hot Plug Detect (HPD) de la señal, una señal de 5V capaz de entregar hasta 50 mA, y un pin reservados (RES). De éstos, sólo los canales de datos diferenciales y el bus I^2C están conectados a la FPGA. Todas las conexiones de señales se muestran en la siguiente figura:



HDMI Type A Connectors

Pin/Signal	J1: IN	J2: Out	J3: IN
1:D2+	B12	B8	J16
2:D2_S	*GND*	*GND*	*GND*
3:D2-	A12	A8	J18
4:D1+	B11	C7	L17
5:D1_S	*GND*	*GND*	*GND*
6:D1-	A11	A7	L18
7:D0+	G9	D8	K17
8:D0_S	*GND*	*GND*	*GND*
9:D0-	F9	C8	K18
10:Clk+	D11	B6	H17
11:Clk_S	*GND*	*GND*	*GND*
12:Clk-	C11	A6	H18
13:CEC	*NC*	*0K to Gnd*	*NC*
14:RES	*NC*	*NC*	*NC*
15:SCL	C13	D9	M16
16:SDA	A13	C9	M18
17:Gnd	*GND*	*GND*	*GND*
18:5V	*JP4**	*5V*	*JP8**
19:HPD	*1K to 5V*	*NC*	*1K to 5V*

*jumper can disconnect Vdd

HDMI Type D

Pin/Signal	JA: BiDi
1:HPD	*JP3**
2:RES	*VCCB2*
3:D2+	N5
4:D2_S	*GND*
5:D2-	P6
6:D1+	T4
7:D1_S	*GND*
8:D1-	V4
9:D0+	R3
10:D0_S	*GND*
11:D0-	T3
12:Clk+	T9
13:Clk_S	*GND*
14:Clk-	V9
15:CEC	*VCCB2*
16:Gnd	*GND*
17:SCL	C13**
18:SCA	A13**
19:5V	*JP3*

**shared with J1 I2C signals via jumper JP2

Figura 16: Conexionado de los puertos HDMI de la placa Atlys

3.2.7 Audio

La placa Atlys incluye un Semiconductor LM4550 AC '97 códec de audio (IC19) con cuatro conectores 1/8" de audio para salida de sonido (J16), una salida para cascos (J18), una entrada de sonido (J15) y una entrada de micrófono (J17). Soporta datos de audio de hasta 18 bits y 48 kHz de ancho de banda, pudiendo tener diferentes anchos de banda para la entrada de audio (grabación) y la salida de audio (reproducción). Excepto el conector para micrófono, que es mono, el resto de conectores son estéreo. La toma de auriculares es alimentada por el amplificador interno de 50mW. La siguiente tabla resume las señales de audio:

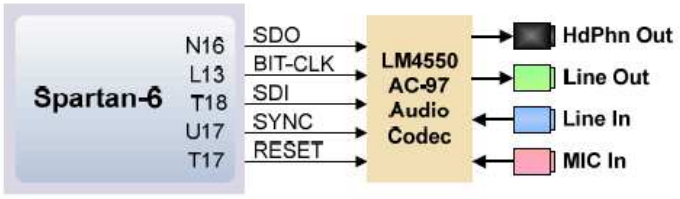


Figura 17: Esquema de puertos de Audio en la placa Atlys

En resumen, la FPGA Atlys Spartan 6 esta específicamente diseñada para el procesamiento de video en tiempo real, lo cual es el objetivo de nuestro proyecto.

3.3 CAMARA ESTEREO

3.3.1 Introducción

La cámara estéreo VmodCAM [2] es un modulo adicional de Atlys que proporciona capacidades de imagen digital a cualquier FPGA de Digilent que tenga un conector VHDCI. Cuenta con dos sensores de imagen digital CMOS Aptina MT9D112 de 2-megapixel cada uno y completamente independientes, los cuales pueden proporcionan imágenes desde 15 FPS (Fotogramas por segundo) en adelante, dependiendo de la resolución, la cual es configurable y va desde 640x480 hasta

1600x1200 pixeles.

El diseño del sistema integra un procesador de flujo de imágenes lo cual permite formatos de salida seleccionables, ajustes de proporcionalidad y efectos especiales. El PLL integrado (phase-locked loop, bucle de enganche de fase) y el microprocesador ofrecen una interfaz de control flexible. En cuanto a los datos de salida, estos se envían en un bus paralelo procesados en formato YCrCb, RGB o Bayer.

Las principales características de la cámara incluyen:

• Dos sensores de imagen digital CMOS Aptina MT9D112 de 2 mega píxeles cada uno y completamente independientes entre si

• Una Resolución máxima de 1600x1200 a 15 fps

• 63 mm de espacio entre las cámaras (línea de base estéreo)

• 10-bit de profundidad de color crudo

• Un Bus de control I^2C

• Formatos de salida Bayer, RGB, YCrCb

• Exposición automática, ganancia y balance de blancos

• Potentes algoritmos de corrección de imagen

• Escalado de imagen

• Memoria intermedia con salida FIFO

• Un conector hembra VHDCI de 68 pines.

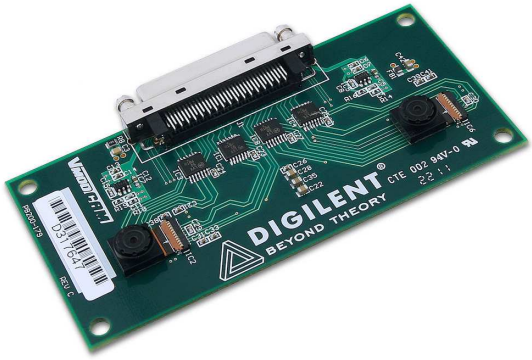


Figura 18: VmodCAM Estéreo

3.3.2 Descripción Funcional

Las dos cámaras MT9D112 se pueden controlar independientemente y pueden adquirir, por separado, dos imágenes de forma simultánea. Ambas cámaras están controladas por una interfaz de dos hilos.

Cada cámara tiene un sensor de color de 2 mega píxeles y dispone de un filtro Bayer. La lectura de los sensores es de 10-bit y soportan saltar o alternar filas y/o columnas durante la captura de la imagen

El PLL integrado puede generar un reloj interno como reloj maestro y es compatible con una amplia gama de resoluciones y frecuencias de imagen.

El procesador de imagen de flujo integrado aplica algoritmos de corrección para mejorar la calidad de imagen y puede procesar datos en bruto provenientes de los sensores para convertirlos a formatos de salida RGB o YCrCb, además de poder escalar la imagen.

Dado que algunos de los datos de algoritmos de tratamiento de salida vienen en ráfagas, la interfaz de salida en paralelo, puede usar una memoria intermedia FIFO para proporcionar una velocidad de datos constante.

La cámara también cuenta con un secuenciador para coordinar eventos activados por el usuario.

3.3.3 Formatos de Salida

En la siguiente tabla se pueden ver los formatos de salida con los que puede trabajar la cámara.

Mode	Byte	D7:D0
RGB565	Odd	$R_7R_6R_5R_4R_3G_7G_6G_5$
	Even	$G_4G_3G_2B_7B_6B_5B_4B_3$
RGB555	Odd	$0R_7R_6R_5R_4R_3G_7G_6$
	Even	$G_5G_4G_3B_7B_6B_5B_4B_3$
RGB444x	Odd	$R_7R_6R_5R_4G_7G_6G_5G_4$
	Even	$B_7B_6B_5B_40000$
RGBx444	Odd	$0000R_7R_6R_5R_4$
	Even	$G_7G_6G_5G_4B_7B_6B_5B_4$
YUV	4*i	Cb
	4*i+1	Y
	4*i+2	Cr
	4*i+3	Y

Tabla 2: Formatos de Salida de la Cámara Estéreo

- El formato RGB565 consiste guardar un pixel en 16 bit, sabiendo que la intensidad de rojo, está definido por los 5 bit más altos, la de verde por los 6 bit intermedios y la de azul por los 5 bit más bajos.

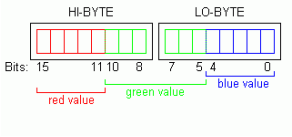


Figura 19: Representación Formato RGB565

- El formato RGB555 es similar al RGB565 con la diferencia de que el bit más alto no se usa, y cada color ocupa 5 bit.

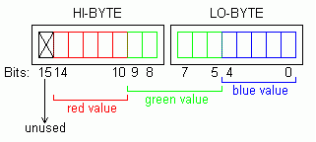


Figura 20: Representación Formato RGB555

- El formato YCrCb (Y Cr Cb), se basa en un modo de transmisión de video con componentes separados que utiliza tres cables diferentes para llevar información con respecto a los componentes de luminancia (luminosidad) y los dos componentes de crominancia (color). El parámetro Y representa la luminancia (es decir, información en blanco y negro), mientras que Cr y Cb representan la crominancia (es decir, información con respecto al color).

3.3.4 Contextos

El microprocesador de la cámara puede usar dos contextos, cada uno con su propio conjunto de parámetros. El contexto A está pre-configurado para el modo de vista previa, mientras que el Contexto B es para instantáneas y captura de vídeo.

Es posible ajustar los parámetros utilizando la interfaz de dos hilos para mandar al secuenciador que cambie contextos. Si los parámetros, que van a ser modificados pertenecen al modo activo, los cambios sólo tendrán efecto después de la actualización y de que se ejecuten los comandos de actualización de modo.

3.3.5 Secuenciador

El secuenciador es una máquina de estado responsable de la coordinación de eventos activados por el usuario. Puede consultar el estado actual y las instrucciones al secuenciador para cambiar de estado.

3.3.6 Resumen

En resumen, la principal ventaja de este modulo, es que ambos sensores pueden ser controlados de forma independiente y ambos pueden adquirir imágenes separadas de forma simultánea, lo cual hace que sea idóneo para realizar sistemas de visión por computador ya que cada sensor, es como un ojo humano, consiguiendo al igual que la visión humana, captar dos imágenes de la misma escena pero con un desplazamiento horizontal entre ambas**.**

El formato que nosotros utilizaremos será RGB565, en el cual la intensidad de rojo, está definido por 5 bit, la de verde por 6 bit y la de azul 5 bit.

La resolución elegida variará en función del diseño; empelaremos la resolución de 1600x1200 para el diseño de HD, y la resolución de 640x480 para el diseño de más baja resolución y el diseño de Mapa de Disparidad y el diseño generador de Anáglifos.

3.4 ENTORNO DE TRABAJO

3.4.1 Introducción

El entorno de trabajo que se ha utilizado para este proyecto ha sido principalmente el paquete informático ISE® de XILINX Inc. Aunque también se ha utilizado el programa Adept Digilent.

3.4.2 Xilinx-ISE

La herramienta Xilinx-ISE (Integrated Software Environment) es una herramienta de software creada por Xilinx para la sintetización y el análisis de diseños creados en HDL (Hardware Descripción Languaje, Lenguaje de descripción de Hardware).

Esta herramienta permite al desarrollador sintetizar ("compilar") sus diseños, realizar análisis de tiempo, examinar diagramas RTL (register-transfer level), simular el comportamiento de un diseño en distintas situaciones y configurar y programar el dispositivo de destino.

Este ha sido el software utilizado para diseñar todas las funcionalidades de nuestro proyecto.

A continuación podemos ver la entidad principal de nuestro diseño

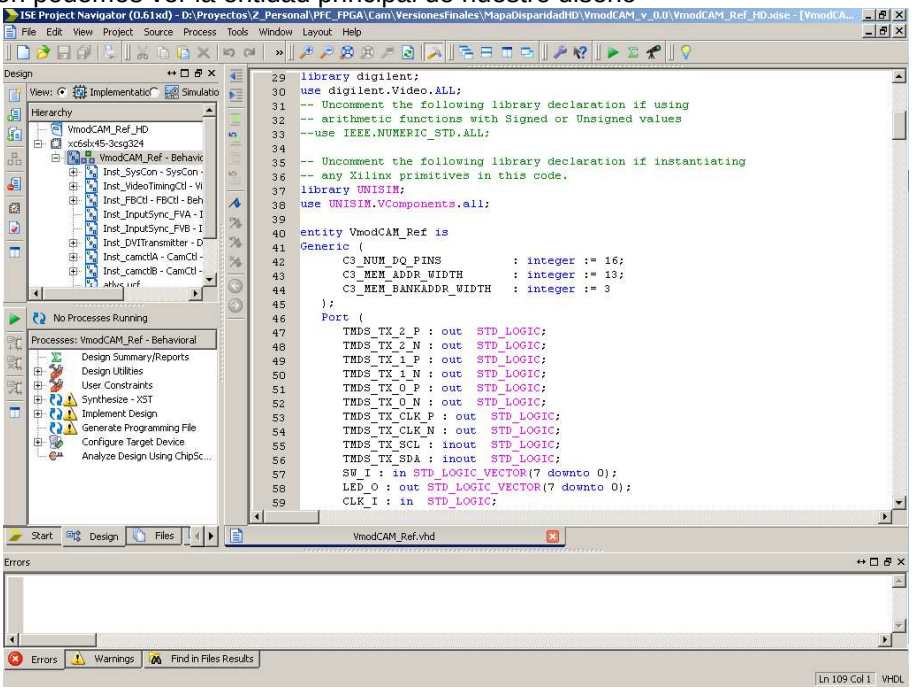


Figura 21: Diseño del bloque principal mediante Xilinx

3.4.3 Adept Digilent

El software Adept Digilent se ha empleado en el proyecto para testear la placa Atlys y programar la FPGA Xilinx Spartan-6 LX45.

3.4.3.1 Programación

Tras generar mediante la herramienta Xilinx-ISE el archivo *.bit, deberemos programar la placa, bien de forma directa mediante el puerto JTAG o a través de la memoria SPI ROM para ello los pasos a seguir son los siguientes:

1. Iniciar el software Adept

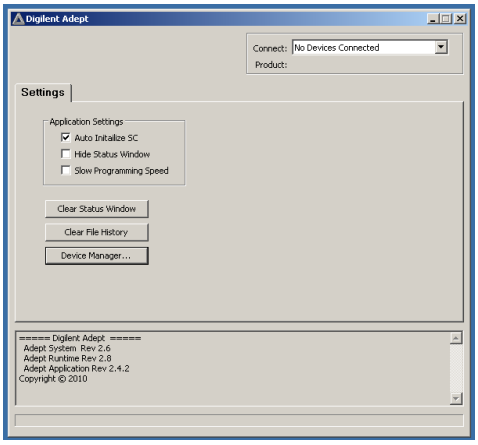


Figura 22: Pantalla inicial del software Adept

2. Enchufar y encender la placa.
3. Seleccionar el dispositivo con el que vamos a conectar

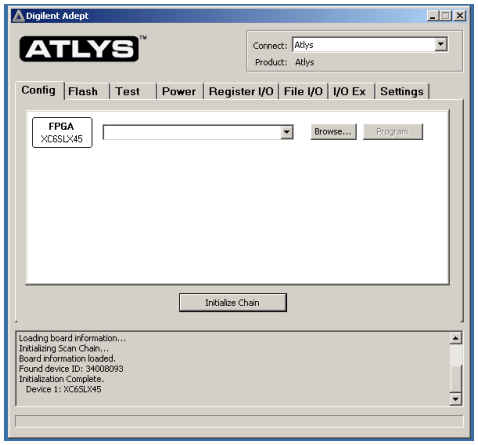


Figura 23: Pantalla tras la selección del dispositivo a conectar

4. Tras seleccionar el dispositivo, deberemos seleccionar como programaremos la FPGA, mediante puerto JTAG, en la pestaña “Config” o a través de la memoria SPI ROM en la pestaña “Flash”; una vez decidida la forma de programar la placa se debe examinar el archivo *.bit, que previamente habíamos generado mediante la herramienta Xilinx-ISE.

5. Tras seleccionar el archivo, podremos programar la placa.

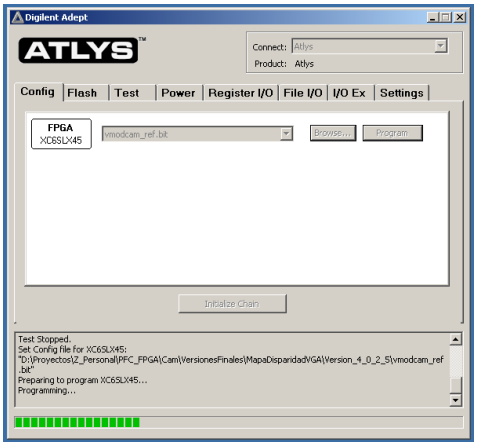


Figura 24: Pantalla durante la programación de la placa

3.4.3.2 Testeo de la plataforma

El software Adept, no solo sirve para programar la placa, sino que además nos permite comprobar el correcto funcionamiento de todas las características de la plataforma Xilinx Spartan-6 LX45 FPGA.

1. En la pestaña “Test” podremos comprobar que tanto los switches como los botones funcionan de forma correcta:

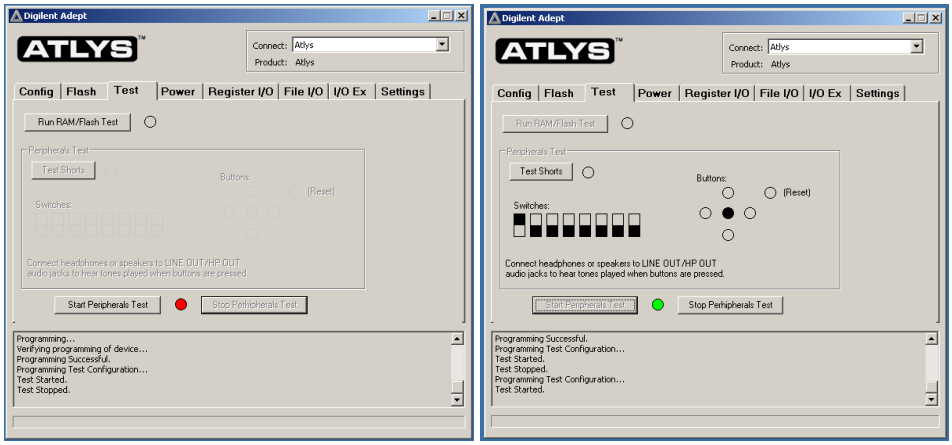


Figura 25: Test de switches y botones

2. En la pestaña “Power”, se puede monitorizar en tiempo real las lecturas de voltaje y corriente de toda la placa:

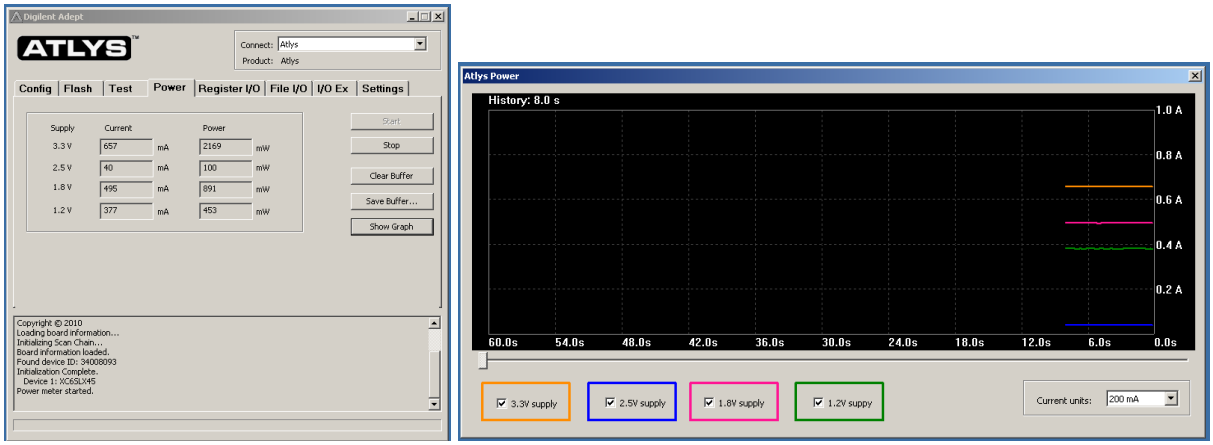


Figura 26: Monitorización de Voltajes y Corrientes

4. ARQUITECTURA DEL DISEÑO

4.1 INTRODUCCION

Este capítulo contiene la descripción del desarrollo de cada uno de los módulos del Proyecto. El modulo principal de nuestro diseño es la interfaz VmodCAM.

4.2 VMODCAM_INTERFACE

Con el fin de interconectar la cámara VmodCAM con la FPGA y con la placa Atlys, es necesario desarrollar un modulo. Este modulo ha sido cogido de la pagina web de Digilent [2] y modificado para adaptarse a nuestras aplicaciones.

El diseño básico de Digilent utiliza varios componentes adicionales a la placa Atlys: memoria DDR2, el puerto de salida HDMI y switches. En la página web había dos diseños, uno con una interfaz que utiliza las cámaras en modo HD con la resolución máxima (1600 x 1200) y otro utilizando las cámaras en modo VGA con la resolución más baja (640 x 480). La resolución VGA, al tener menor número de pixeles, ofrece un mayor número de fotogramas por segundo, por lo que se determinó que la resolución del proyecto fuera esta, para conseguir una mayor velocidad de ejecución.

Cuando el proyecto con resolución VGA, es programado en la placa Atlys, se muestra a través del puerto de salida HDMI la imagen que captan los sensores de la cámara. Con el fin de implementar futuros diseños, se ha tenido que entender la función de cada uno de los módulos de los que consta el diseño básico.

En ambos diseños básicos primero se captura la imagen (en forma de serie de pixeles) desde la cámara utilizando el modulo “TWI_Ctl”. Este modulo es el responsable de utilizar la interfaz I^2C para comunicarse con la cámara. La salida de este modulo sería un pixel (16 bits) proveniente de la cámara. De forma predeterminada, los pixeles de 16 bits están en formato RGB565, esto significa que los 5 bits superiores se corresponden con el color rojo, los 6 bits intermedios corresponden al verde y los 5 bits más bajos corresponden al azul. El siguiente paso en el proceso es almacenar el pixel en la memoria DDR2 de la placa, para posteriormente utilizando el modulo “DVITransmitter” mostrar la imagen a través del puerto de salida HDMI. El siguiente diagrama de bloques muestra el flujo básico de los pixeles en el código de Digilent:

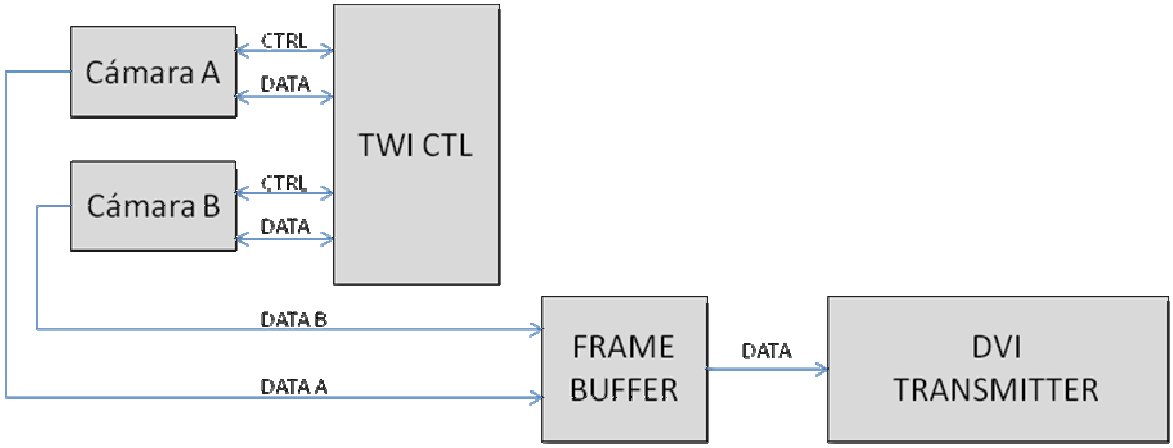


Figura 27: Diagrama de Bloques del Código de Digilent

En el diagrama anterior, se muestran los cuatro grandes bloques necesarios para implementar el diseño, aunque hay algunos módulos más como es el Syscon y el VideoTimingCtl. Todos ellos se describen a continuación.

4.2.1 TWI_Ctl Module

4.2.1.1 Introducción

Este componente es el responsable de proporcionar la interfaz I^2C para comunicar con la cámara. El controlador puede ordenar iniciar, detener o continuar la transferencia de datos utilizando las señales STB_I y MSG_I. La gestión del flujo de datos es proporcionada por la señal DONE_O y ERR_O. Las señales de salida del modulo son síncronas al reloj, CLK, y las señales de entrada también deben ser síncronas a este reloj, además todas las señales son activas a nivel alto.

4.2.1.2 Proceso TWI_Ctl

El proceso para la captura de una imagen desde la cámara requiere varios pasos que se describen a continuación:

1. Cuando la señal DONE_O está a '0', el modulo está preparado para aceptar comandos de transferencia de datos.
2. Las transferencias de datos puede ser iniciada poniendo la dirección del esclavo TWI en el bus A_I y activando la señal de luz estroboscópica, STB_I.
    a. La dirección de los datos a transferir (lectura/escritura) es determinada por el LSB (*Least Significant Bit*, Bit Menos Significativo) de la dirección, siendo '0' para escribir y '1' para leer.
        i. Si la transferencia es una escritura, los datos deben aparecer en el bus D_I antes de que la entrada de luz estroboscópica, STB_I, se active.
3. Una vez que los datos se han leído ó escrito, la señal DONE_I se activa durante un ciclo de reloj.
    a. Si la señal DONE_I = '1' y la señal ERR_O = '0' significa que la operación se ha realizado con éxito.
    b. En cambio, si la señal ERR_O = '0' y la señal DONE_I también está a '1', la operación ha resultado errónea.
4. Después de la activación de la señal DONE_I, hay un periodo de tiempo, concretamente ¼ del periodo que del modulo TWI, en el que puede ser enviada la siguiente luz estroboscópica, lo cual es muy útil se envían o reciben varios bytes en un paquete de transferencia único.
    a. Si no se proporciona una nueva luz estroboscópica, y por tanto la señal STB_I permanece a 0, la transferencia se da por terminada.
    b. Si se proporciona una nueva luz estroboscópica, pero la dirección de transferencia cambia, la actual transferencia de datos se dará por terminada y comenzara una nueva.
5. El inicio de una nueva transferencia de datos se puede forzar mediante la activación de la

señal MSG_I a '1'. La ventaja de esto solo es relevante en los buses con varios maestros, ya que en lugar de esperar a que la transferencia termine, se puede liberar e iniciar una nueva transferencia son renunciar al control sobre el bus de datos.

La salida del componente TWI son los pixeles capturados por las cámaras. Estos datos se almacenan en los módulos CamCtlA y CamCtlB, respectivamente.

4.2.2 CamCtlA y CamCtlB

4.2.2.1 Introducción

Para poder trabajar con el sistema de la cámara, este primero debe estar correctamente configurado; esto incluye, no sólo establecer los parámetros de imagen, como la resolución o el formato de salida, sino también la configuración PLL y del microprocesador de secuenciación. El orden en el que estos pasos se realizan es muy importante de cara a un correcto funcionamiento del sistema.

Es en el modulo CamCtl donde se programan estas configuraciones; este modulo, proporcionan una sencilla interfaz para la lectura de los datos de video, haciendo de enlace entre el modulo TWI_Ctl Module y el resto de módulos, ya que la salida de este módulo es un vector de 16 bit, 1 pixel.

Para configurar las cámaras, en primer lugar, se debe configurar el encendido y la secuencia de reinicio, para luego entender la interfaz de control y poder configurar las características de resolución, formato de salida, etc. En las secciones siguientes se describen en detalle.

4.2.2.2 Descripción del modulo CamCtl

4.2.2.2.1 ENCENDIDO Y SECUENCIA DE RESTABLECIMIENTO

La VmodCAM sólo debe ser conectado a la placa del sistema una vez que las señales controladas por la placa del sistema se definen. La cámara utiliza los carriles de suministro de voltaje analógicas y digitales proporcionadas por la placa. Las fuentes de alimentación están activadas por defecto, pero puede desactivarse, controlando que la señal VDD-EN esté bajo.

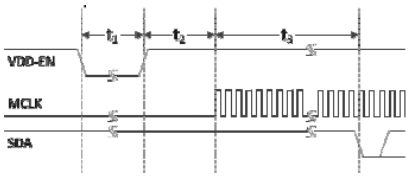


Figura 28: Secuencia de Encendido

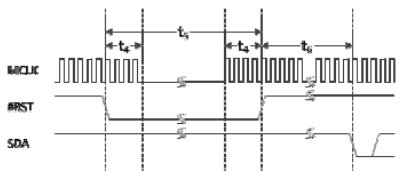


Figura 29: Secuencia de Reinicio

Las fuentes de alimentación son utilizadas por ambas cámaras. Aunque por sí mismas las cámaras hacen un reinicio justo después del encendido, siempre es una buena idea hacer un reinicio completo como parte de la rutina del controlador.

El MCLK es importante. Si la señal del PLL en la cámara está activada, la señal MCLK debe ser estable.

Detener la señal MCLK sin respetar la secuencia de restablecimiento podría dejar la cámara en un estado indefinido. Este podría producirse durante la reconfiguración de la de la FPGA, por lo que se recomienda realizar un ciclo de encendido al principio del funcionamiento.

4.2.2.2.2 INTERFAZ DE CONTROL

La interfaz serie de dos hilos (SDA, SCL) se puede utilizar para controlar diversas partes de la cámara. La cámara actúa como un dispositivo esclavo.

- Una escritura normal de un registro típico se compone de:
  - Condición de inicio
  - Dirección 8-bit dispositivo (0x78 para la MT9D112) + bit de reconocimiento
  - Byte superior del registro de direcciones de 16-bit + bit de reconocimiento
  - Byte más bajo de la dirección de registro + bit de reconocimiento
  - Byte superior de los datos de 16-bits + bit de reconocimiento
  - Menor byte de datos + bit de reconocimiento
  - Parada.
- Una lectura normal de un registro típico se compone de:
  - Condición de inicio
  - Dirección 8-bit dispositivo (0x78 para la MT9D112) + bit de reconocimiento
  - Byte superior del registro de direcciones de 16-bit + bit de reconocimiento
  - Byte más bajo de la dirección de registro + bit de reconocimiento
  - Condición de inicio
  - Dirección 8-bit dispositivo (0x79 para la MT9D112) + bit de reconocimiento
  - Byte superior de los datos de 16-bits + bit de reconocimiento
  - Menor byte de datos + bit de no reconocimiento
  - Parada.

Hay dos tipos de controles para los registros del hardware y las variables del controlador. Los registros de hardware normalmente controlan el sensor y algunos otros subsistemas, mientras que las variables del controlador secuencian el chip del microprocesador. El acceso a estos dos tipos de controles se realiza de forma diferente.

- Los registros de hardware son accesibles mediante dos hilos, es decir, su dirección puede ser utilizada directamente en la fase de registro. Los registros de hardware son direccionados

directamente por su dirección. Por ejemplo, R [0x3000] se refiere al registro ubicado en la dirección 0x3000.

- A las variables del controlador se puede acceder directamente a través de dos registros de hardware, R [0x338C] y R [0x3390]. Para acceder a una variable, su primera dirección necesita ser escrita a R [0x338C], que es un estándar escritura en el registro de dos hilos. Luego, la lectura de registro R [0x3390] lee la variable y escribe su valor en ella, por lo que establece la variable con ese valor. Las variables del controlador son direccionadas por su dirección. Por ejemplo, V [0x2797] se refiere a la variable de controlador que se encuentra en la dirección 0x2797.
-

4.2.2.2.3 CONFIGURACION

Las cámaras arrancan con unos valores por defecto guardados en los registros, lo cual significa que están en espera. Para obtener imágenes de las cámaras, ambas tienen que ser inicializadas correctamente. A medida que se realiza el encendido y se muestran secuencias de reinicio, la cámara necesita ciertas señales establecidas y una ejecución de la señal MCLK antes de que se active la interfaz de dos hilos.

Una vez que son proporcionados el número necesario de ciclos de MCLK, los registros y variables que se mencionan a continuación deben ser leídos y/ó escritos.

- **Identificar la cámara:**

  Para comprobar que la cámara funciona, se debe leer el registro R [0x3000] el cual debe valer 0x1580, que es el dispositivo de ID de la cámara.

- **Reinicio de MCU:**

  Para realizar y finalizar el reinicio del MCU se deben escribir los siguientes registros:

  1. R[0x3386] = 0x0501, reinicio
  2. R[0x3386] = 0x0500 fin del reinicio.

- **Seleccionar el PLL:**

  Para la configuración del PLL se deben escribir los siguientes registros:

  1. R [0x3214] = 0x0D85, lo que establece la velocidad de subida de los pines de salida.
  2. R [0x341E] = 0x8F0B, apagado y evitar PLL. Si desea utilizar MCLK como el reloj de píxeles, se deben omitir los pasos 3-5.
  3. R [0x341C] [13:08] = N

     R [0x341C] [7:0] = M

     Donde PCLK = MCLK * M / (N +1) / 8. Por ejemplo, para obtener un reloj de píxel de 80 MHz (máximo) de una MCLK de 24MHz, elija M = 80, N = 2.
  4. R [0x341E] = 0x8F09; enciende el PLL y espera 1 ms para el PLL para estabilizar
  5. R [0x341E] = 0x8F08, usar reloj PLL en lugar de MCLK

- **Activar tras la espera:**

  Se debe escribir el siguiente registro:

  1. R[0x3202] = 0x0008

- **Parámetros de la Imagen**

  A continuación se muestra un ejemplo de la secuencia de inicialización para configurar la cámara para una salida de 16-bit RGB565 y una resolución de 1600x1200 en modo de captura de vídeo. Estos parámetros son variables del controlador y se puede acceder como se describe en la sección Interfaz de Control vista anteriormente. Todos los pasos son de escritura de variable salvo el número 11, que es de lectura. En el anexo 1, se pueden ver todos los parámetros de imagen.

  1. V [0x2797] = 0x0030; Formato de Salida; contexto B y formato RGB565
  2. V [0x2707] = 0x0640; ancho de salida 1600
  3. V [0x2709] = 0x04B0, altura de salida 1200
  4. V [0x275F] = 0x0000; Crop X0 0
  5. V [0x2763] = 0x0000; Crop Y0 0
  6. V [0x2761] = 0x0640; Crop X1 1600
  7. V [0x2765] = 0x04B0; Crop X1 1200
  8. V [0x2741] = 0x0169; Auto exposición / ganancia fija
  9. V [0xA120] = 0x00F2; Balance de blancos automático, exposición automática, Autohistograma, Video en modo Captura
  10. V [0xA103] = 0x0002; Modo de Actualización secuencial
  11. V [0xA103] = 0x0000, se debe leer la variable hasta que sea igual a 0x0000, lo cual significaría que el comando ya se habría ejecutado.

- **Habilitar Salida:**

  Escribir el siguiente registro:

  1. [R] = 0x301A 0x02CC; habilitar salida paralela, comenzar la transmisión.

4.2.2.2.4 PROCESAMIENTO DE IMÁGENES

Los datos sin procesar del sensor procedentes de la matriz de píxeles alimentan un procesador de flujo de la imagen (IFP). Aquí es donde todo el procesamiento de la imagen, algoritmos de corrección, escalado, interpolación y formato de salida se aplican.

El IFP también se puede omitir, haciendo que la salida de la cámara sean 10 bit, sin comprimir en formato Bayer.

El IFP es controlado indirectamente, a través de variables del microprocesador y del secuenciador.

El IFP se puede controlar directamente mediante el acceso a los registros del hardware aunque normalmente será el microprocesador el que ajuste estos parámetros.

- **Secuenciador**

  El secuenciador es una máquina de estado responsable de la coordinación de eventos activados por el usuario. Puede consultar el estado actual y las instrucciones al secuenciador para cambiar de estado. La variable de comando secuenciador V [0xA103] acepta los siguientes valores:

  • 0-Run

  • 1- Vista previa

  • 2-Capturar

  • 3-Espera

  • 4-Bloquear

  • 5-Actualizar

  • 6-Actualizar el modo

  Para la captura de video, el bit de Video (bit 1) tiene que establecerse en la variable V [0xA120]. De lo contrario, la estado de captura sólo tendrá una sola instantánea.

  El secuenciador de estado lee la variable V [0xA104]:

  • 0-Inicializar

  • 1-Cambiar el modo de Vista previa

  • 2-Introducir Vista Previa

  • 3-Vista previa

  • 4-Dejar Prevista

  • 5-Cambiar el modo de Captura

  • 6-Introducir Captura

  • 7-Captura

  • 8-Dejar captura

  • 9-Standby

La salida de los módulos CamCtl, que es un vector de 16 bit, o lo que es lo mismo un pixel, se asignan a través de señales internas a los datos de entrada de la memoria interna (Frame Buffer)

4.2.3 FBCtl (Frame Buffer)

4.2.3.1 Introducción

El modulo FBCtl, es el modulo encargado de gestionar tanto la lectura como la escritura de la memoria DDR2 de la FPGA. Este modulo es el encargado de gestionar que la imagen que se capta mediante los módulos CamCtlA y CamCtlB, sea almacenada y posteriormente enviada al puerto HDMI.

4.2.3.2 Descripción del modulo

4.2.3.2.1 ENTIDAD PRINCIPAL

En cuanto a la entidad principal, el modulo FBCtl, tiene 5 grupos diferenciados de entradas/salidas:

1. El primer grupo de son las señales del puerto C, que es el encargado de enviar la imagen desde la memoria hasta el puerto HDMI; las señales de entrada ENC, CLKC y RSTC_I son las encargadas de controlar la lectura de la memoria; la entrada RD_MODE, indica el estado de los switches (véase modulo SysCon) y en el caso de los programas de ejemplo controla que imagen mostrar si la imagen de la Cámara A, de la Cámara B o de las dos. La salida DOC, es el vector donde se guarda el pixel que se envía al puerto HDMI, que como hemos visto en el modulo CAmCtl, ocupa 16 bits, ya que esta en formato RGB565.
2. El segundo grupo son las señales del puerto B encargado de guardar en memoria la imagen proveniente de la cámara B. En este grupo todas las señales son entradas: las entradas ENB, RSTB_I y CLKB controlan la escritura en memoria, mientras que la entrada DIB es el pixel (16 bits) proveniente de la cámara B.
3. El tercer grupo son las señales del puerto A encargado de guardar en memoria la imagen proveniente de la cámara A. En este grupo, al igual que en el anterior, todas las señales son entradas: las entradas ENA, RSTA_I y CLKA controlan la escritura en memoria, mientras que la entrada DIA es el pixel (16 bits) proveniente de la cámara A.
4. El cuarto grupo son las señales de entrada del reloj y la interfaz PLL.
5. El quinto grupo son las señales de control de la interfaz de la memoria DDR2; existen tanto entradas como salidas, así como, señales de entradas/salidas.

4.2.3.2.2 MODULO “mcb_ddr2”

Dentro del modulo FBCtl, se encuentra la llamada al modulo “mcb_ddr2” el cual contienen librería creada por Xilinx para gestionar la lectura/escritura de la memoria DDR2. Este modulo tiene además de las señales de control de reloj e interfaz PLL y señales de interfaz de la memoria DDR2, tiene las señales de los puertos 0, 1, 2 y 3, donde se asignan las señales que escriben y/o leen de la memoria:

- El puerto 0, se puede usar tanto para escribir como para leer la memoria, tiene un grupo de señales las cuales configuran el puerto, que son las señales (p0_cmd…), otro grupo que son las señales para escribir, (p0_wr…) y un último grupo que son las señales para leer de la memoria, (p0_rd…).
- El puerto 1, al igual que el puerto 0, también se puede usar tanto para escribir como para leer

la memoria, y tiene señales de configuración (p1_cmd…), de escritura (p1_wr…) y de lectura (p1_rd…).

- El puerto 2, únicamente se puede usar para escribir en la memoria y por tanto solo tiene un grupo de señales de configuración del puerto (p2_cmd…) y otro grupo para escribir en la memoria (p2_wr…).
- El puerto 3, únicamente se puede usar para leer de la memoria y por tanto solo tiene un grupo de señales de configuración del puerto (p3_cmd…) y otro grupo para leer de la memoria (p3_rd…).

El uso de las principales señales que forman un puerto de lectura y/o escritura, se explica a continuación, utilizando como ejemplo el puerto 0, que puede ser de lectura ó de escritura

- Señales de configuración:
    - p0_cmd_clk, donde se asigna el reloj que controla el puerto.
    - p0_cmd_en, es la señal que indica si se ejecuta el mando (escribir o leer en cada caso) o no.
    - p0_cmd_instr, donde se escribe el uso que tendrá el puerto de lectura o escritura.
    - p0_cmd_bl, donde se escribe el tamaño de lo que se guardara.
    - p0_cmd_byte_addr, la dirección de memoria donde escribir/leer el dato.
    - Además también están las señales p0_cmd_empty y p0_cmd_full.

- Señales de escritura
    - p0_wr_clk, se asigna el reloj que controla el puerto.
    - p0_wr_data, donde se guarda el dato que queremos escribir
    - p0_wr_en, habilita la escritura.
    - p0_wr_empty, indica que el puerto ha sido reseteado y que no hay más datos esperando para ser escritos.
    - p0_wr_count, indica el numero de datos que quedan en el buffer por escribir.
    - Además también están las señales p0_wr_mask, p0_wr_underrun, p0_wr_full y p0_wr_error

- Señales de lectura
    - p0_rd_clk, se asigna el reloj que controla el puerto.
    - p0_rd_en, habilita la lectura
    - p0_rd_data, donde se guarda el dato leído.
    - Además están las señales p0_rd_full, p0_rd_empty, p0_rd_count, p0_rd_overflow, p0_rd_error.

4.2.3.3 Descripción del Funcionamiento

La descripción del funcionamiento del modulo FBCtl, aunque se ha intentado hacer lo más genérica posible, se ha de indicar que se ha utilizado como referente el diseño básico VGA.

En este diseño se emplea el puerto 3 para leer la memoria y enviarla al HDMI, el puerto 1 para escribir la imagen proveniente de la cámara A y el puerto 2 para escribir la imagen de la cámara B, quedando el puerto 0, sin utilizar.

Hay que decir, que a cada una de las cámaras se le asigna un área de memoria, VMEM_SIZE de 2**23 (2^23) bits de memoria, lo cual es equivalente a almacenar una imagen (640*480*16).

A continuación se explican cada uno de los puertos:

4.2.3.3.1 PUERTO C, LECTURA

El puerto 3 del modulo mcb_ddr2, emplea las señales que ENC, CLKC, RSTC_I.

La salida p3_rd_data del modulo mcb_ddr2 es asignada a la salida DOC, que es la salida del FBCtl y que es el pixel leído que ira posteriormente hacia el HDMI.

En este diseño, para seleccionar la dirección de lectura, existe un bucle, que va desde 0 hasta el tamaño de la imagen (640x480x2), y es mediante el estado de los switches 7 y 6 como seleccionamos la dirección:

1. Si el interruptor 7 es ‘1’ y el 6 es ‘0’, se muestra la imagen del sensor A.
2. Si el interruptor 7 es ‘0’ y el 6 es ‘0’, se muestra la imagen del sensor B.
3. Si ambos interruptores son ‘1’ se muestra una pantalla dividida que muestra la mitad de cada una de las dos imágenes.

Es importante destacar en este punto que si se eligiera un diseño con otra resolución, habría que editar el tamaño de la imagen y ajustarlo al nuevo tamaño.

Todo el proceso de lectura está controlado por una maquina de estados, en la cual destaca el estado “stRdCmd” que es cuando se ejecuta el mando de leer de la memoria.

4.2.3.3.2 PUERTO A, ESCRITURA CAM A

El puerto 1 del modulo mcb_ddr2, emplea las señales ENA, CLKA, RSTA_I.

A la entrada p1_wr_data se le asigna la entrada DIA, que es el pixel proveniente de la cámara A, pero como DIA ocupa 16 bits, y p1_wr_data es de 32 bits, DIA se asigna tanto a la parte alta como a la parte baja; es por esto que en el tamaño de la imagen, en el proceso de selección de la dirección de memoria, es multiplicado por 2.

Para indicar la dirección de memoria de escritura, existe un bucle desde 0 hasta el tamaño de la imagen (640*480*2).

Al igual que el puerto C, todo el proceso de escritura de la imagen de la cámara A, es controlado por otra máquina de estados.

4.2.3.3.3 PUERTO B, ESCRITURA CAM B

Es un proceso similar al de la escritura de la cámara A; aquí, el puerto 2 del modulo mcb_ddr2, emplea las señales ENB, CLKB, RSTB_I.

A la entrada p2_wr_data se le asigna la entrada DIB que es el pixel proveniente de la cámara B, y al igual que con la cámara B, como DIB ocupa 16 bits, y p2_wr_data es de 32 bits, DIB se asigna tanto a la parte alta como a la parte baja; es por esto que en el tamaño de la imagen, durante el proceso de selección de la dirección de memoria, es multiplicado por 2.

Para indicar la dirección de memoria de escritura, existe un bucle desde 0 hasta el tamaño de la imagen (640*480*2), al que luego se le suma VMEM_SIZE, para escribirlo en una de memoria distinta a la de la cámara A.

Al igual que el puerto C y puerto A, todo el proceso de escritura de la imagen de la cámara A, es controlado por otra máquina de estados.

Un vez que los datos son leídos de la memoria, mediante el puerto C, mediante señales internas, se envían al modulo DVITranmitter para mostrarlos por pantalla a través del puerto HDMI.

4.2.4 DVI_Transmitter

4.2.4.1 Introducción

El modulo DVITransmitter, es el modulo encargado de enviar los datos de video al puerto de salida HDMI para que sean mostrados por pantalla codificando y serializando los datos de acuerdo al formato de Interfaz Visual Digital (DVI).

4.2.4.2 Descripción del modulo

El modulo DVITransmitter tiene como señales de entrada principales 3 vectores de 8 bit cada uno, que son el valor de rojo (Red), verde (Green) y azul (Blue) que forman un pixel en formato RGB; además tiene entradas de sincronización de video y de reloj y salidas de control de video.

4.2.4.3 Descripción del Funcionamiento

El componente DVITransmitter codifica, mediante señales de sincronización, los 24-bit de un pixel de vídeo en formato RGB, en un dato de 30-bits, es decir, 10 bit para cada color, mediante la llamada a la librería “TMDSEncoder”; estad codificación conocida como TMDS (Transition Minimized Differential Signaling, Señal Diferencial de Transición Minimizada) es una tecnología de transmisión de datos en serie a alta velocidad y utilizada por las interfaces de vídeo DVI y HDMI, en la que un proceso de dos etapas convierte una entrada de 8 bits en un código de 10 bits con propiedades especiales deseables.

Tras tener codificado el pixel, el componente serializa la señal de video y la sincroniza mediante las señales de entrada de reloj vistas anteriormente.

4.2.5 SysCon

4.2.5.1 Introducción

Este componente es el encargado de proporciona un reloj al sistema, un reset síncrono y otras señales necesarias para todo el sistema, entre los que destacan el estado de los interruptores.

4.2.5.2 Descripción del Modulo

En este componente como es lógico, casi todas las señales son de salida, y entre todas ellas destacan:

- RSEL_O, que se utiliza en el modulo FBCtl, para indicar que zona de memoria leer, es decir, , en el caso del diseño VGA, es el estado de los switches 7 y 6.
- PCLK_O, PCLK_X2_O y PCLK_X10_O, que son las señales de control utilizadas por el modulo DVITransmitter para serializar el video, ya que proporciona soporte para la reconfiguración dinámica, lo que permite el cambio en la frecuencia de reloj y del formato de vídeo.
- DDR2CLK_2X_O y DDR2CLK_2X_180_O, utilizadas en el modulo FBCtl, para controlar el acceso a la memoria DDR2.
- PLL_CE_0_O, PLL_CE_90_O y PLL_LOCK, que proporcionan las señales de control del PLL.

4.2.6 VideoTimingCtl

4.2.6.1 Introducción

El componente VideoTimingCtl es el encargado de generar las señales de sincronización vertical y horizontal adecuadas.

4.2.6.2 Descripción del Modulo

El modulo VideoTimingCtl tiene como entrada principal el valor de la resolución del diseño, como por ejemplo R640_480P, ya que de este valor dependen las señales de sincronización; además tiene las entradas de reset y reloj del sistema.

Como salidas tiene la habilitación para poner datos en el bus de pixeles, así como, las señales de control para la sincronización vertical y horizontal.

4.2.7 VmodCAM_Ref

4.2.7.1 Introducción

Es el componente principal de todo el diseño, en este componente están las entradas y salidas de la placa y las llamadas al resto de componentes; es en el modulo VmodCAM_Ref, donde mediante señales internas se conectan las entradas de unos módulos con las salidas de otros y viceversa.

4.2.7.2 Descripción

- Llamada al modulo Inst_SysCon: se llama a este modulo para conseguir las señales de reset y de reloj del sistema, así como otras muchas señales, vistas en el modulo SysCon, que luego serán entradas de otros módulos
- Llamada al modulo Inst_VideoTimingCtl: se llama a este modulo para conseguir las señales de sincronización de video; en la llamada hay que escribir la resolución con la que se trabajará en el diseño.
- Llamada al modulo Inst_FBCtl: para escribir y leer la memoria; las tres señales principales son las entradas DIA y DIB, a las que se leas asignan las señales que contienen los pixeles provenientes de los módulos CamCtlA y CamCtlB, respectivamente, y la salida DOC, que contiene el pixel leído, y que posteriormente se asignará al modulo DVITransmitter.
- La llamada al modulo Inst_DVITransmitter, para trasmitir el pixel leído al puerto HDMI; como entradas, además de las ya mencionadas señales de sincronización vertical y horizontal, tiene el pixel que se va a enviar al puerto HDMI, y como vimos en la descripción de este modulo, a este entran 3 señales de 8 bits cada una, con el valor de rojo, verde y azul, es decir un pixel en formato RGB; sin embargo, en el modulo FBCtl la salida es un pixel en formato RGB565, por lo que hay antes de llamar al modulo, hay que hacer una conversión e formatos, de la siguiente manera: el valor rojo, que se almacena en los bits 15 a 11, se concatena con tres ‘0’ para así conseguir los 8 bits; el valor verde, que se almacena en los bits 10 a 5, se concatena con dos ‘0’; y el valor azul, que se almacena en los bits 4 a 0, se concatena con tres ‘0’. De esta manera hemos convertido un pixel en formato RGB565 a formato RGB.
- Llamada al modulo CamCtl: esta llamada hay que realizara dos veces, una con las configuración y señales de la cámara A, Inst_camctlA y otra con la configuración y señales de la cámara B, Inst_camctlB.

4.3 MAPA DE DISPARIDAD

4.3.1 Introducción

En este capítulo se describe cómo será el algoritmo con el que construiremos el Mapa de Disparidad.

4.3.2 Calculo del Mapa de Disparidad

Como se comentó anteriormente, el mapa de disparidad es una imagen cuyos niveles de gris indican la distancia horizontal, en píxeles, entre las proyecciones de un punto del mundo, en cada uno de los planos de imagen de cada cámara.

A modo de simplificación, se suponen las cámaras calibradas y las imágenes rectificadas, de modo que la búsqueda se reduce a una línea horizontal (línea epipolar) y por tanto, la coordenada *y* es común a ambas imágenes.

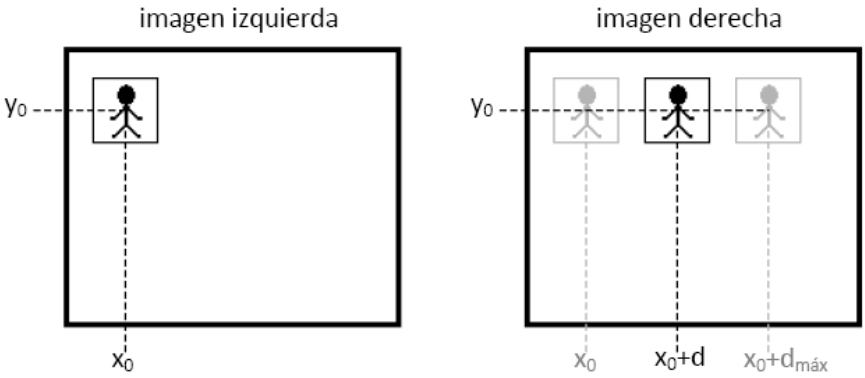


Figura 30: Proceso de comparación entre 2 imágenes estéreo

Tomando una ventana de la imagen izquierda como referencia, se busca en la imagen derecha, la ventana que ofrezca mayor similitud. Para una ventana de la imagen izquierda centrada en el píxel (x0,y0)izda, la búsqueda comienza en la misma posición de la imagen derecha (x0,y0)dcha prolongándose hasta (x0+dmáx,yo)dcha, donde dmáx es la disparidad máxima (figura 10).

Para analizar la similitud entre dos píxeles se ha elegido como función de coste, en base a su mayor facilidad de implementación, el método de las diferencias absolutas de las intensidades. El valor resultante se nombra como AD (*Absolute Difference*). Por su parte, la suma de los AD de todos los píxeles que componen la ventana, correspondiente a la etapa de agregación de coste, se nombra como coste o SAD (*Sum of Absolute Differences*).

En el vector coste se almacenan los valores del coste de comparar cada par de ventanas (x0,y0)izda con (x0,y0)dcha y posteriores hasta (x0+dmáx,yo)dcha. Siguiendo el cálculo de la disparidad mediante el método local WTA, una vez hallados todos los valores de este vector coste, se comparan en busca del menor, cuya posición marca la disparidad del píxel estudiado, (x0, y0).

En este algoritmo, para el cálculo de la SAD entre dos ventanas se requiere comparar
Uno a uno todos los píxeles que las componen. Esto se basaría en un bucle de la forma:

```
for j in 0 to n loop
        for i in 0 to n loop
                coste <= coste + abs (im_izda[j, i] - im_izda[j, i])
        end loop
end loop
```

Este bucle debe ir anidado a otros dos, habiendo en total tres bucles anidados entre sí, quedando en algoritmo de la siguiente manera, vistos los bucles del más interior al más exterior:

1. El bucle más pequeño, visto anteriormente, recorre todos los elementos de una ventana para el cálculo de la SAD.
2. El segundo bucle, permite comparar las ventanas izquierdas con las ventanas derechas y posteriores.
3. El tercer bucle, recorre todos los pixeles de la imagen izquierda. Desde la ventana superior izquierda, se desplaza en cada columna de arriba abajo y de izquierda a derecha.

4.4 ANAGLIFO

4.4.1 Introducción

En este capítulo se explicara cómo se crea y como funciona un anáglifo, que recordemos que es una imagen de dos dimensiones capaz de provocar un efecto tridimensional, cuando se ve con lentes especiales.

4.4.2 Crear Anáglifos

Para crear un anáglifo es primordial que ambas imágenes sean tomadas en el mismo momento, y en las mismas condiciones de luz y escenografía; además, en nuestro caso, como cada imagen es captada por cámaras diferentes, ambas cámaras deben tener la misma configuración de resolución, ganancia, etc.

Tras captar las imágenes se aplicara el algoritmo en sí:

1. La imagen que representa al ojo izquierdo, únicamente debe dejar pasar el color rojo.
2. La imagen que representa al ojo derecho, solo debe dejar pasar los colores verde y rojo.

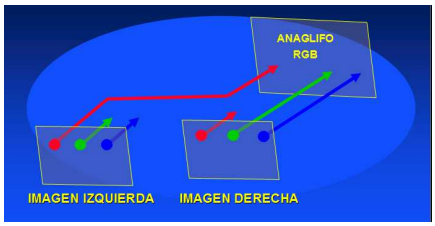


Figura 31: Esquema de Generación de Anáglifos

3. Finalmente se superponen una imagen sobre la otra

Figura 32: Ejemplo de un anáglifo [9]

4.4.3 Funcionamiento

Ver anáglifos a través de filtros de color apropiados da como resultado que cada ojo observa una imagen levemente diferente. En un anáglifo rojo-azul (más exactamente, cian, que es el complementario del rojo) por ejemplo, el ojo cubierto por el filtro rojo ve las partes rojas de la imagen como "blancas" y las partes azules como "oscuras" (el cerebro produce la adaptación de los colores).

Por otro lado, el ojo cubierto por el filtro azul percibe el efecto opuesto. El resto de la composición son percibidas iguales por los ojos. El cerebro fusiona las imágenes recibidas de cada ojo, y las interpreta como una imagen con profundidad.

4.5 IMPLEMENTACION DEL DISEÑO COMPLETO

En este apartado se describirán cada uno de los diseños finales que se han implementado.

4.5.1 Diseño Básico VGA

Es el diseño básico conseguido en la página web de Digilent, cuyas principales características específicas son que está preparado únicamente para visualizar la imagen que captan los sensores, utilizando una resolución de 640 x 480 y seleccionando que imagen se mostrar por el HDMI mediante los switches 7 y 6.

A continuación se puede ver un esquema del diseño:

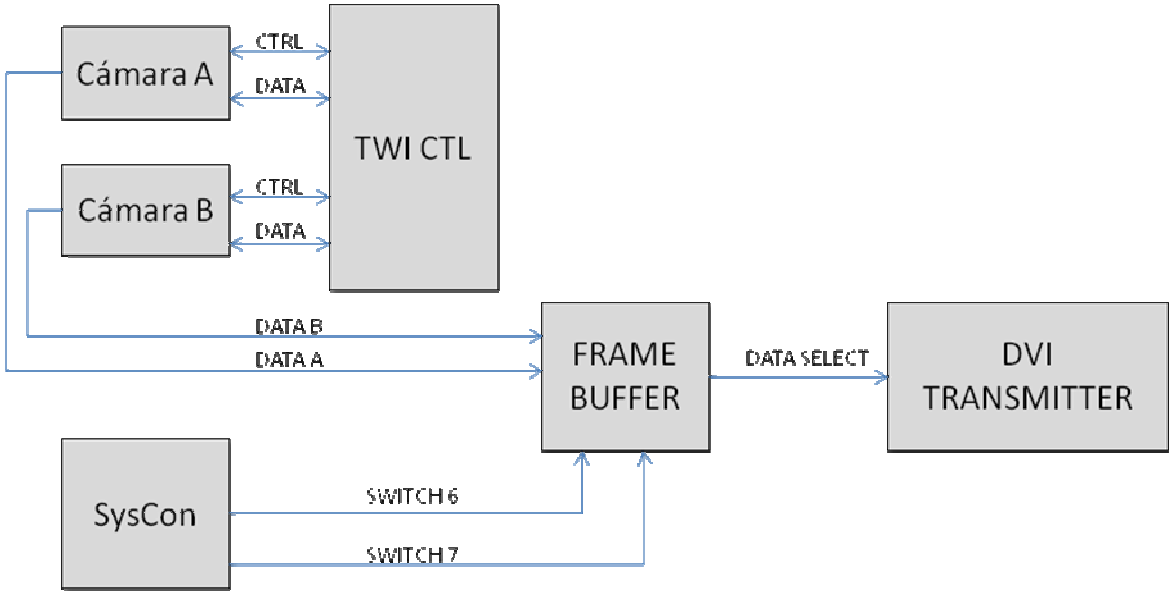


Figura 33: Diagrama de Bloques del Diseño básico VGA

Para configurar una resolución de 640 x 480, no solo hay que configurar la resolución de las cámaras, además hay otras muchas señales que configurar para un correcto funcionamiento:

- CamCtl: como vimos en capítulos anteriores, para configurar las cámaras hay que escribir los valores adecuados en unas variables determinadas; además hay que tener en cuenta que este diseño funcionara en el Contexto A, en modo “vista previa”; para el caso concreto de la resolución, la configuración seria:

```
IWR & x"338C2703", -- Ancho de salida, para el contexto A
IWR & x"33900280", -- 640dec, (280hex)
IWR & x"338C2705", -- Altura de salida, para el contexto A
IWR & x"339001E0", -- 480dec, (1E0hex)
```

- FBCtl, en este fichero se ha de tener en cuenta la resolución para determinar los limites de los bucles utilizados para seleccionar la dirección de memoria a la que acceder; a continuación, podemos ver el bucle utilizado para seleccionar la dirección de escritura de la cámara A:

```
if (pa_wr_addr = 640*480*2/(WR_BATCH*4)-1) then
        pa_wr_addr <= 0;
else
        pa_wr_addr <= pa_wr_addr + 1;
end if;
```

- SysCon, en este fichero se debe indicar la resolución del diseño, para poder configurar la frecuencia de funcionamiento del sistema:

```
--------------------------------------------------------------------------
-- Dynamic resolution change
--------------------------------------------------------------------------
res <= R640_480P;
process(res)
begin
        case (res) is
                when R640_480P => --640x480@60Hz = 25MHz
```

- Finalmente, en la entidad principal en VmodCAM_Ref, donde se ajusta la resolución a la entrada del modulo Inst_VideoTimingCtl, para general las señales de sincronización vertical y horizontal

```
Inst_VideoTimingCtl: entity digilent.VideoTimingCtl PORT MAP (
                PCLK_I => PClk,
                RSEL_I => R640_480P, --this project supports only VGA
```

En cuanto al uso de los switches para seleccionar que imagen mostrar a través del HDMI, en este diseño al utilizar únicamente dos switches la señal de lectura de estos es de 2 bit; en el modulo VmodCAMRef, esta señal que conecta la salida del modulo SysCon, donde se lee, con la entrada del modulo FBCtl, donde se usa, es la señal *MSel*. En el modulo FBCtl, esta señal se llama RD_MODE, y es usada para seleccionar que zona de memoria leer mediante el puerto C, para posteriormente enviar al HDMI. En este diseño, tal y como se ve a continuación, se puede mostrar la imagen A, la imagen B o la mitad de ambas a la vez:

1. Si el interruptor 7 es ‘1’ y el 6 es ‘0’, se muestra la imagen del sensor A.
2. Si el interruptor 7 es ‘0’ y el 6 es ‘1’, se muestra la imagen del sensor B.
3. Si ambos interruptores son ‘1’ se muestra una pantalla dividida que muestra la mitad de cada una de las dos imágenes.

Todo este proceso, se puede ver en el modulo FBCtl, en el proceso:

```
--------------------------------------------------------------------
-- Read Addressing
--------------------------------------------------------------------
        RDADDRCNT_PROC: process (CLKC)
```

4.5.2 Diseño Básico HD

Es el diseño básico conseguido en la página web de Digilent, cuyas principales características específicas son que funcionan en modo captura de video, que emplea una resolución de 1600 x 1200 y que emplea el switch 7 para seleccionar que imagen mostrar por el HDMI.

A continuación se puede ver un esquema del diseño:

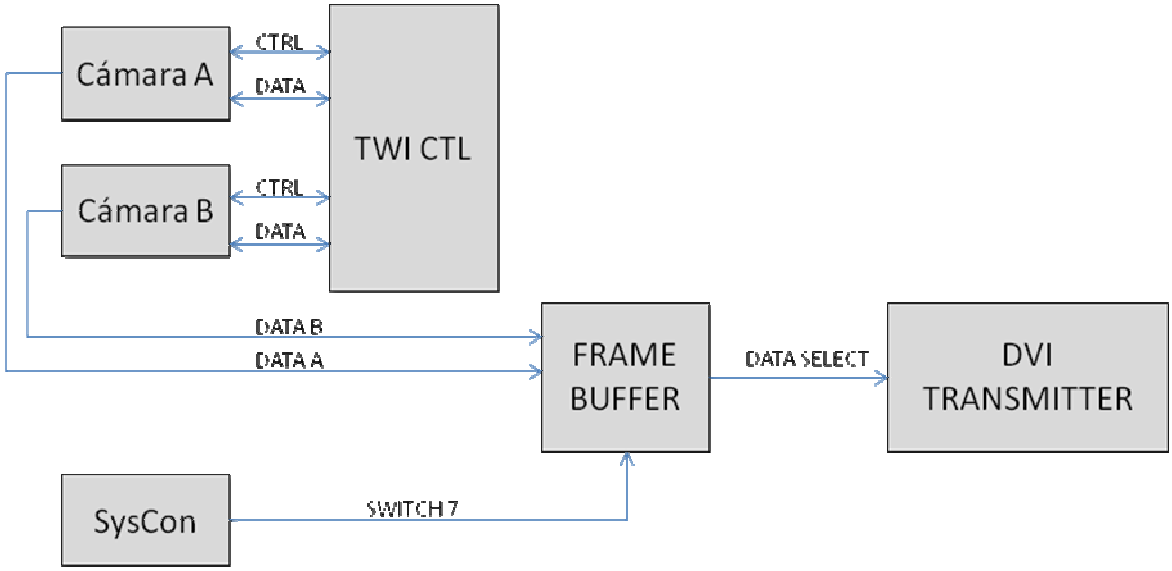


Figura 34: Diagrama de Bloques del Diseño básico HD

Como se ha visto, en el diseño anterior, para configurar una resolución específica, no solo hay que configurar la resolución de las cámaras, además hay otras muchas señales que configurar para un correcto funcionamiento:

- CamCtl: en este caso para configurar una resolución de 1600 x 1200, teniendo en cuenta que trabajaremos con el contexto B, "captura de vídeo", la configuración sería:

```
IWR & x"338C2707", -- Ancho de salida, para el contexto B
IWR & x"33900640", -- 1600dec (640hex)
IWR & x"338C2709", -- Altura de salida, para el contexto B
IWR & x"339004B0", -- 1200dec (4B0hex)
```

- FBCtl, en este fichero se ha de tener en cuenta la resolución para determinar los limites de los bucles utilizados para seleccionar la dirección de memoria a la que acceder; a continuación, podemos ver el bucle utilizado para seleccionar la dirección de escritura de la cámara B:

```
if (pb_wr_addr = 1600*1200*2/(WR_BATCH*4)-1) then
        pb_wr_addr <= 0;
else
        pb_wr_addr <= pb_wr_addr + 1;
end if;
```

- SysCon, en este fichero se debe indicar la resolución del diseño, para poder configurar la frecuencia de funcionamiento del sistema, sin embargo aunque la resolución de las cámaras es de 1600x1200, la resolución esta recortada hasta 1600x900, para poder ajustarse a la resolución de la pantalla:

```
------------------------------------------------------------------------
-- Dynamic resolution change
------------------------------------------------------------------------
res <= R1600_900P;
```

```
process(res)
begin
        case (res) is
                when R1600_900P =>=> --1600x900@60Hz = 108MHz
```

- Finalmente, en la entidad principal en VmodCAM_Ref, donde se ajusta la resolución a la entrada del modulo Inst_VideoTimingCtl, para general las señales de sincronización vertical y horizontal, ocurriendo lo mismo, que en el caso de la frecuencia, donde se ajusta la resolución para trabajar

```
Inst_VideoTimingCtl: entity digilent.VideoTimingCtl PORT MAP (
                PCLK_I => PClk,
                RSEL_I => R1600_900P, --this project supports only 1600x900
```

En cuanto al uso de los switches para seleccionar que imagen mostrar a través del HDMI, en este diseño al utilizar únicamente un switches la señal de lectura es de 1 solo bit; al igual que en el diseño, básico de VGA, la señal que conecta la salida del modulo SysCon, donde se lee, con la entrada del modulo FBCtl, donde se usa, en el modulo VmodCAMRef, es la señal *MSel*, con la diferencia de que esta vez es un vector de 1 solo bit. En el modulo FBCtl, esta señal se llama RD_MODE, y es usada para seleccionar que zona de memoria leer mediante el puerto C, para posteriormente enviar al HDMI. En este diseño, tal y como se ve a continuación, solo se puede mostrar o bien la imagen A, o bien la imagen:

1. Si el interruptor 7 es ‘0’, se muestra la imagen del sensor A.
2. Si el interruptor 7 es ‘1’, se muestra la imagen del sensor B.

Todo este proceso, se puede ver en el modulo FBCtl, en el proceso:

```
----------------------------------------------------------------------
-- Read Addressing
----------------------------------------------------------------------
        RDADDRCNT_PROC: process (CLKC)
```

4.5.3 Diseño Para el Cálculo Mapa de Disparidad

En este diseño se pretendía, usando uno de los diseños básicos conseguidos en la página web de Digilent crear un diseño que muestre a través de HDMI el mapa de disparidad de las imágenes que captaran los sensores de la cámara, ya que como se mencionó anteriormente, el mapa de disparidad es una imagen cuyos niveles de gris indican la distancia horizontal, en píxeles, entre las proyecciones de un punto del mundo en cada uno de los planos de imagen de cada cámara, y por tanto nuestra cámara VmodCAM, es ideal para este cometido, ya que de forma instantánea consigue imágenes del mismo entorno y con las mismas características, desplazadas entre sí unos centímetros.

Tras estudiar ambos diseños básicos, se decidió usar como código base el Diseño Básico VGA, ya que al tener menor resolución (640 x 480), el numero de iteraciones que tendría que hacer el algoritmo, sería mucho menor y por lo tanto menos costoso en tiempo.

Tras estudiar y comprender el código base, se concluyó que el lugar para implementar el algoritmo de mapa de disparidad, era en el modulo FBCtl, ya que era donde se tenía acceso a la imagen completa, quedando el diseño completo de Cálculo de Mapa de Disparidad de la siguiente manera:

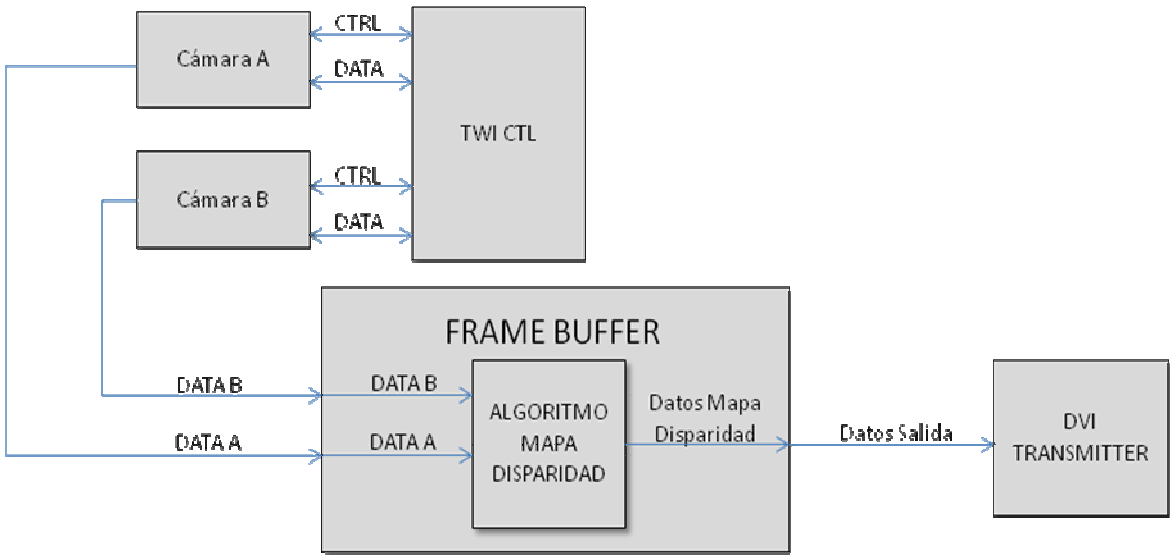


Figura 35: Diagrama de Bloques del Diseño Generador Mapas de Disparidad

La idea inicial era capturar una imagen completa y guardarla en la memoria, posteriormente aplicar el algoritmo de cálculo de mapa de disparidad y finalmente, leer la memoria hacia el HDMI.

Lo primero que había que hacer era transformar una imagen en formato de color RGB565 a tonos grises, ya que la esencia del mapa de disparidad es encontrar distancias en función del tono de gris; para ello, antes de escribir los pixeles en la memoria, se aplicaba el siguiente algoritmo de conversión:

```
Red_A <= conv_integer (DIA(15 downto 11));
Green_A <= conv_integer (DIA(10 downto 5));
Blue_A <= conv_integer (DIA(4 downto 0));

Suma_A <= (Red_A*8*30/100) + (Green_A*4*59/100) + (Blue_A*8*11/100);

Pixel_A(15 downto 11) <= conv_std_logic_vector(Suma_A/8,5);
Pixel_A(10 downto 5) <= conv_std_logic_vector(Suma_A/4,6);
Pixel_A(4 downto 0) <= conv_std_logic_vector(Suma_A/8,5);

p1_wr_data(15 downto 0) <= Pixel_A;
```

El algoritmo, primero convertía a entero cada uno de los tonos de color que formaban un pixel, posteriormente los pasaba a RGB y aplicaba algoritmo para pasar a tonos grises, lo pasaba de nuevo a formato de vector y finalmente lo guardaba en la variable que se escribía en la memoria.

En cuanto al tratamiento de la imagen, para tener un control total de cuando se tenía una imagen completa y que no comenzara guardarse una nueva, se decidido que cada vez que se guardara una imagen, se parara la escritura de otra nueva, congelando la imagen, hasta que el algoritmo de mapa de disparidad hubiera terminado, ya que se entendía que debido al gran número de operaciones que tenía que hacer el algoritmo, este iría más lento que la escritura de la imagen. En un principio esto se controlaba manualmente mediante el switch 6, cuando se ponía a 0, y además se tenía una imagen completa, se congelaba la escritura; indicar que una imagen completa se tenía, cuando la dirección de escritura alcanzaba el limite, como se ve a continuación:

```
if (pa_wr_addr = 640*480*2/(WR_BATCH*4)-1) then
        pa_wr_addr <= 0;
        if (RD_MODE(0) = '1') then
                guardar <= '1';
        else
                guardar <= '0';
        end if;
else
        pa_wr_addr <= pa_wr_addr + 1;
end if;
```

Resuelto el problema del retardo, el siguiente problema que apareció fue el no acceso directo a la memoria; al contrario que, por ejemplo en lenguaje, c, donde generalmente una imagen es tratada como una matriz numérica, donde cada valor de la matriz es un pixel distinto, en nuestro diseño esto no ocurría así, ya que como se vio en la descripción del modulo FBCtl, para tratar la memoria, ya sea escritura ó lectura, había que hacerlo mediante unos puertos determinados lo cual dificultaba el diseño; para solucionar este problema se decidió que el uso de los puertos fuera el siguiente:

- Puerto 0 (Lectura/Escritura): se usaría para escribir la imagen, primero de A, y después de B; gracias a la velocidad de la FPGA, el hecho de escribir primero una imagen, y después la otra, no afectaría al diseño, y como se veía durante los pruebas de cada paso, era algo imperceptible, al ojo humano.
- Puerto 1 (Lectura/Escritura): se usaría para leer de la memoria durante la ejecución del algoritmo de mapa de disparidad.
- Puerto 2 (Escritura): se usaría para escribir en memoria el mapa de disparidad.
- Puerto 3 (Lectura): se usaría para leer los datos del Mapa de disparidad hacia el HDMI

Para todo ello se delimitaron 3 zonas en la memoria:

- Imagen A: 0 - VMEM_SIZE.
- Imagen B: VMEM_SIZE - 2*VMEM_SIZE.
- Mapa de Disparidad: 2*VMEM_SIZE - 3*VMEM_SIZE.

Tras definir todo lo anterior, y para probar que lo diseñado era viable, se creó un diseño que escribiría la imagen A, mediante el puerto 0, posteriormente la leería mediante el puerto 1, la imagen sería tratada para que solo se mostrara en tonos rojos, y escrita mediante el puerto 2, en la zona de memoria del Mapa de disparidad, y finalmente mediante el switch 7, se elegiría que imagen mostrar, bien la imagen guardada en la zona de memoria para la imagen A, o la zona de memoria del Mapa de Disparidad, sin embargo los resultados y alguna pruebas posteriores, demostraron que esto no funcionaba de forma correcta debido a que el proceso de lectura no estaba sincronizado con el de escritura, con lo cual se perdían pixeles, lo que hacía que se mostraran pixeles aleatorios y sin ningún sentido.

Tras muchos intentos y más horas de estudio y comprensión del diseño básico, se resolvió que tal y como estaba realizado el código base no era posible realizar esto ya que para poder realizar lo diseñado de forma teórica habría que rehacer prácticamente por completo, todo el código base referente a la escritura y lectura de la memoria, ya que además se necesitarían nuevas señales procedentes del modulo SysCon, que sincronizaran los procesos.

4.5.4 Diseño Para la Generación de Anáglifos

En este diseño, al igual que en el anterior, se pretendía que usando uno de los diseños básicos conseguidos en la página web de Digilent, conseguir un sistema, que permitiera ver un entorno de forma tridimensional, ya que las imágenes anáglifo han vuelto a despertar el interés general, no solo en la industria del entretenimiento, como cine o videojuegos, sino también para la ciencia, donde en muchas investigaciones la percepción de profundidad es útil; un ejemplo claro, es el de la NASA, la cual usa dos vehículos orbitales para obtener imágenes en 3D del Sol.

Tal y como se explico de forma teórica en capítulos anteriores, la técnica anáglifo consiste en crear un efecto tridimensional, mediante dos fotos de un mismo entorno y en las mismas condiciones, desplazadas entre si entre 3 y 5 cm, y aunque la separación de los sensores en nuestra cámara es de 63 mm, y sobresale algo de los márgenes ideales, al igual que para el diseño de mapa de disparidad, la cámara VmodCAM es ideal para este diseño.

Para este diseño, se ha elegido el Diseño Básico VGA, ya que al tener dos switches activos, daba la posibilidad, de mostrar, la pantalla en anáglifo, o solo la imagen A, o solo la imagen B.

El diagrama de bloques del diseño anáglifo es el siguiente:

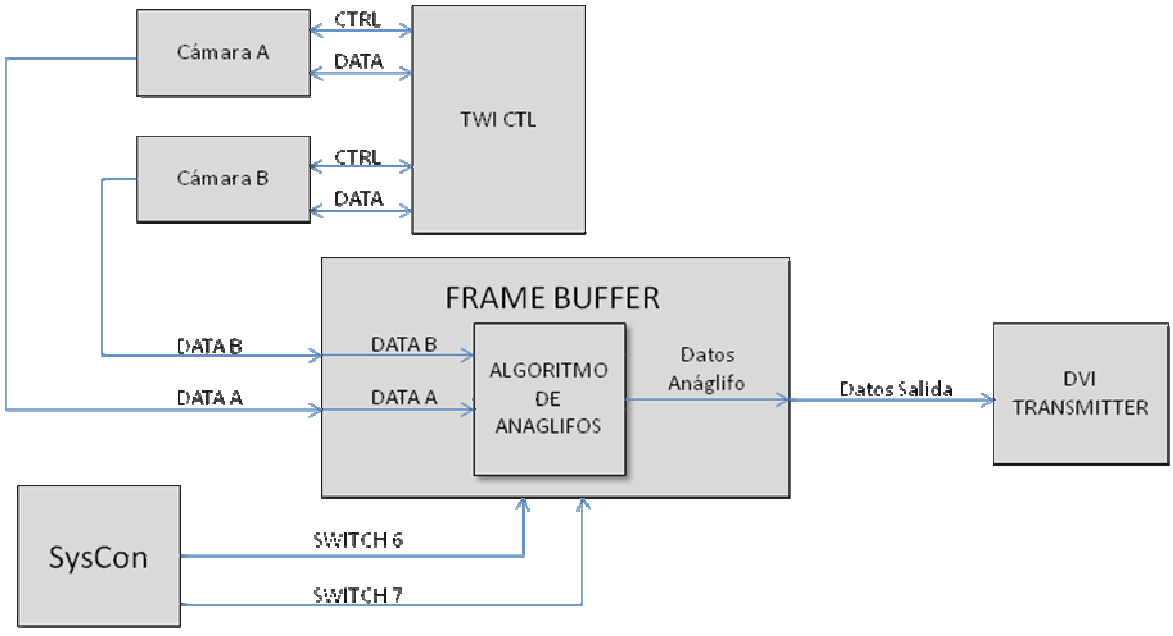


Figura 36: Esquema Diseño Generador Anáglifos

Tal y como se vio, en la explicación de los Anáglifos, para crear anáglifos, hay que superponer dos imágenes, en las cuales, la que represente el ojo izquierdo únicamente deje pasar el rojo, y que la que represente el ojo derecho, deje pasar solo el azul y el verde.

Tras estudiar el diseño básico VGA, al igual que en el diseño de Mapa de Disparidad, se decidió que el lugar adecuado para introducir los cambios adecuados, para conseguir el objetivo fuera el modulo

FBCtl.

Lo primero que había que hacer era identificar, dentro del código, que imagen, si la A o la B, representaba el ojo derecho y cual el ojo izquierdo, y tras algunas pruebas se determinó, tal y como se ve en el siguiente esquema, que la imagen A, representa al ojo izquierdo y la imagen B, al ojo derecho.

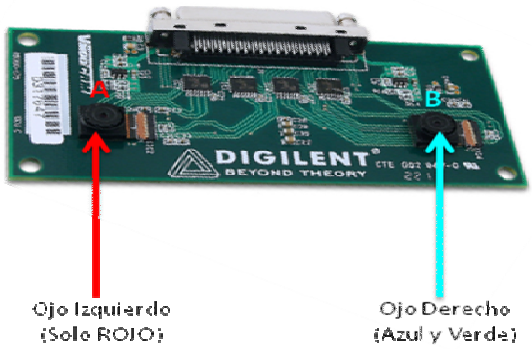


Figura 37: Esquema de la Cámara Estéreo

Tras identificar cada una de las cámaras, el siguiente paso, era conseguir, que la imagen A, solo utilizara el color rojo, y que la imagen B, solo utilizara los colores azul y verde; para esto, era fundamente tener en cuenta que el diseño utilizaba el formato de color RGB565, que como se vio anteriormente, consiste en guardar un pixel en un vector de 16 bit, sabiendo que la intensidad de rojo, está definido por los 5 bit más altos, la de verde por los 6 bit intermedios y la de azul por los 5 bit más bajos, por tanto para la imagen A habría que aislar los bit 15 a 11y para la imagen B los bit 10 a 0.

Una vez aislados los colores correspondientes a cada ojo, el siguiente paso es superponer ambas imágenes una sobre la otra; como se vio en la descripción del diseño básico VGA, la selección de qué imagen mostrar por el HDMI, se realizaba mediante los switches 7 y 6, eligiendo mostrar una u otra, o la mitad de ambas, pero no ambas superpuestas. Para resolver este problema se pensaron dos soluciones:

1. Alternar en pantalla la imagen A y la imagen B.
2. Entrelazar los pixeles de la Imagen A y de la Imagen B, es decir, que la imagen que se muestre, esté formada por ambas imágenes, por ejemplo, que los pixeles pares sean de la imagen A, y los impares de la imagen B.

Finalmente y tras observar los resultados, nos decantamos por la opción 1, la de alternar ambas imágenes.

Todo lo mencionado anteriormente, tanto el aislamiento de cada color como la alternancia de las imágenes, se realizo, para facilitar el entendimiento del diseño, tras leer de la memoria por el puerto C, y justo antes de enviar al HDMI, añadiendo un proceso intermedio, lo cual se puede ver a continuación:

En el siguiente código cada vez que se lee una imagen entera, se leen los switch 7 y 6:

- Si ambos están a 0 o a 1, se niega la variable "int_rd_mode(0)" para saber que se ha cambiado de imagen;
- Si 7 = 0, y 6 = 1, "int_rd_mode(0)" = 1, y por tanto se muestra la imagen B.
- Si 7 = 1, y 6 = 0, "int_rd_mode(0)" = 0, y por tanto se muestra la imagen A.

```
if (pc_rd_addr1 = 640*2*480/(RD_BATCH*4)-1) then
        pc_rd_addr1 <= 0;
        pc_rd_addr2 <= VMEM_SIZE/(RD_BATCH*4);

        if (RD_MODE = "11") then
                int_rd_mode(0) <= not int_rd_mode(0); --anáglifo
        elsif (RD_MODE = "00") then
                int_rd_mode(0) <= not int_rd_mode(0); --anáglifo
        elsif (RD_MODE = "01") then
                int_rd_mode(0) <= '1'; -- azul
        elsif (RD_MODE = "10") then
                int_rd_mode(0) <= '0';          --rojo
        end if;
else
        pc_rd_addr1 <= pc_rd_addr1 + 1;
        pc_rd_addr2 <= pc_rd_addr2 + 1;
end if;
```

En el siguiente proceso antes de guardar en la señal "DOC" que es la salida del modulo FBCtl, se aíslan los pixeles rojos, para la imagen A (int_rd_mode(0) = '0'), o verdes y azul, para la imagen B, (int_rd_mode(0) = '1').

```
Anaglifo: process (CLKC)
begin
        DOC <= (others => '0');
        if int_rd_mode(0) = '0' then
                DOC (15 downto 11) <= DOC_aux (15 downto 11);
        else
                DOC (10 downto 0) <= DOC_aux (10 downto 0);
        end if;
end process;
```

5. RESULTADOS

5.1 INTRODUCCIÓN

En este capítulo se presentaran los resultados obtenidos de cada uno de los diseños, presentando y explicando algunas imágenes. Recordemos, que los diseños que estudiamos fueron el diseño básico VGA, el diseño básico HD, el diseño de Cálculo de Mapa de Disparidad y el diseño de Generación de Anáglifos; de los 4 diseños mencionados, se mostraran resultados de ambos diseños básicos y del diseño de Generación de Anáglifos, ya que como se revolvió, en el capitulo 4.5.3, el diseño para el cálculo de Mapas de Disparidad no pudo ser llevado a cabo, debido a que requería rehacer prácticamente por completo el código base.

Se ha de indicar que muchas de las imágenes que se muestran como resultados son fotografías de la pantalla de una TV.

Para comenzar, mostraremos una imagen de la FPGA con cada una de las conexiones necesarias en todos los diseño.

- A la izquierda, se puede ver el cable JTAG, mediante el cual se programa la FPGA desde el PC
- En la parte superior se pueden ver el cable de alimentación y la salida HDMI
- En la parte derecha, se puede ver el cable, que mediante conexión VHDC, conecta la placa con la cámara estero.
- En la parte inferior podemos ver los switches, que luego intervienen en el diseño.

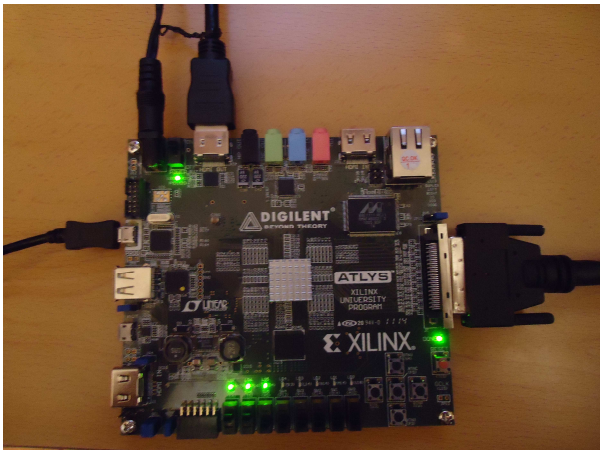


Figura 38: Conexiones de la FPGA durante los Test

A continuación, se muestra la cámara estero ya conectada al cable con conexión VHDC:

Figura 39: Cámara Estéreo conectada al cable VHDC

5.2 DISEÑO BÁSICO VGA

En este apartado se pueden ver imágenes de los resultados que se obtienen cargando en la FPGA el diseño básico VGA de Digilent.

Recordemos que en el diseño básico VGA, el cual utilizaba una resolución de 640x480, se podía mostrar a través de la salida HDMI, y dependiendo de los switches 7 y 6, la imagen captada por el sensor A (ojo izquierdo), la imagen captada por el sensor B (ojo derecho) o la mitad de cada una de las dos imágenes:

- Si el switch 7 está a '1' y el 6 está a '0', se muestra la imagen captada por el sensor A.

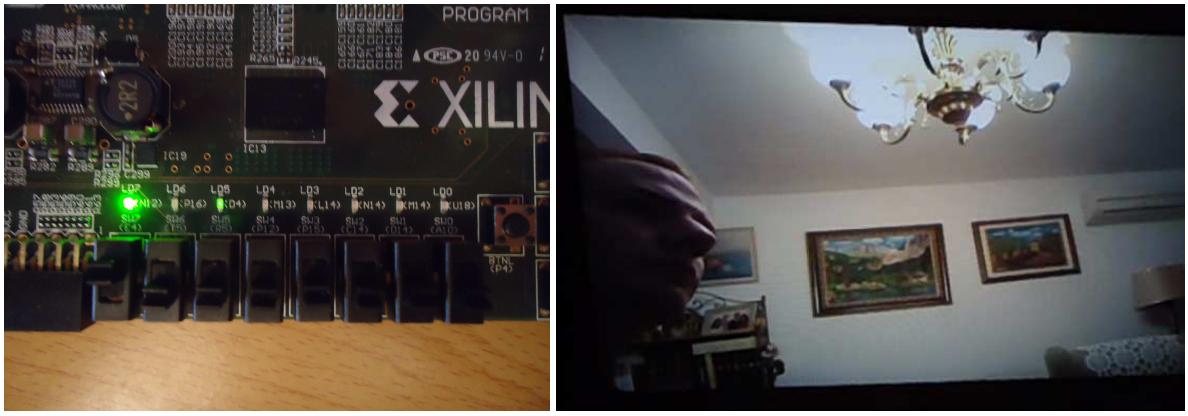


Figura 40: Estado de los Swtich 7('1') y 6('0') e Imagen del Sensor A

- Si el switch 7 está a '0' y el 6 está a '1', se muestra la imagen captada por el sensor B.

Figura 41: Estado de los Swtich 7 (‘0’) y 6 (‘1’) e Imagen del Sensor B

- Si ambos switches se encuentran a ‘1’, se muestra la mitad de la cada una de las dos imágenes.

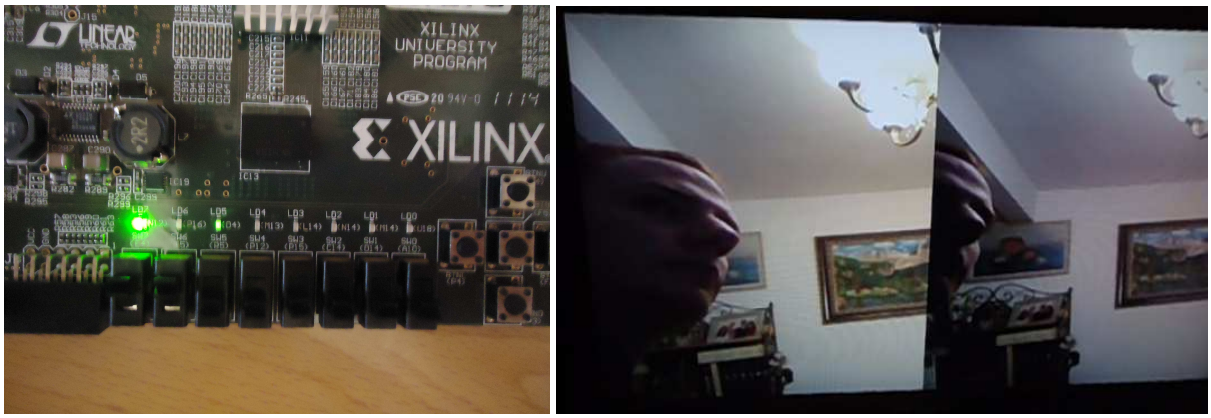


Figura 42: Estado de los Swtich 7 (‘1’) y 6 (‘1’) e Imagen de Ambos Sensores

En la secuencia de imágenes anterior, se puede observar como aunque las dos primeras imágenes parecen iguales, son diferentes en los laterales, igual que si una persona viera una sala primero solo con el ojo izquierdo y después solo con el ojo derecho; en la última imagen podemos ver claramente las diferencias entre ambas imágenes, ya que en la imagen del ojo izquierdo se ve prácticamente el rostro completo, mientras que en la de la derecha, únicamente se ve la mitad del rostro.

5.3 DISEÑO BÁSICO HD

En este apartado se pueden ver imágenes de los resultados que se obtienen cargando en la FPGA el diseño básico HD de Digilent.

Recordemos que en el diseño básico HD, que utiliza una resolución de 1600x1200, se podía mostrar a través de la salida HDMI, y dependiendo del switch 7, la imagen captada por el sensor A (ojo izquierdo) o la imagen captada por el sensor B (ojo derecho):

- Si el switch 7 está a ‘0’, se muestra la imagen captada por el sensor A.

Figura 43: Estado del Swtich 7 (‘0’) e Imagen del Sensor A

- Si el switch 7 está a ‘1’, se muestra la imagen captada por el sensor B.

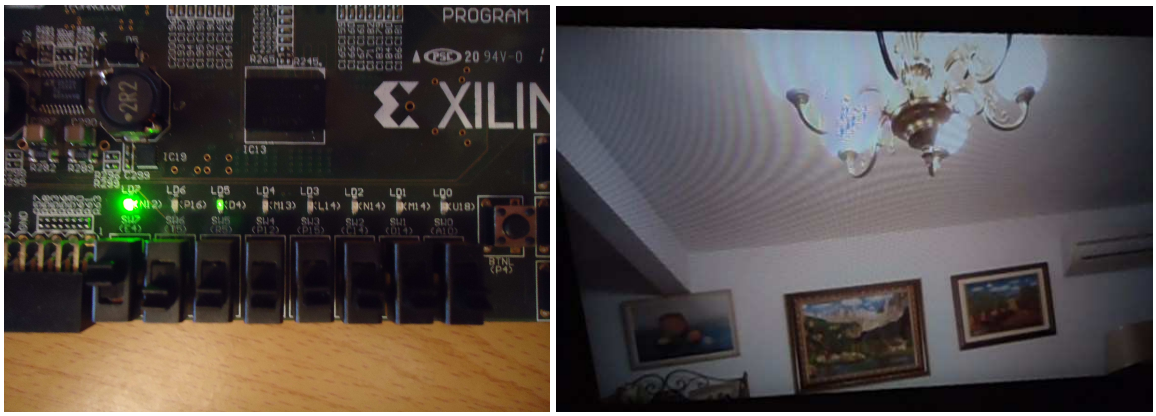


Figura 44: Estado del Swtich 7 (‘1’) e Imagen del Sensor B

Al igual que en el diseño básico, la primera imagen (imagen A) simula lo que vería una persona solo con el ojo izquierdo y segunda imagen, lo que vería un persona solo con el derecho.

5.4 DISEÑO ANÁGLIFO

Por último, en este apartado mostraremos los resultados que se obtienen cargando en la FPGA el diseño para la Generación de Anáglifos.

Recordemos que en este diseño, que trabaja con una resolución de 640 x 480, se generan anáglifos en tiempo real del entorno que este delante de los sensores. Como se explico en capítulos anteriores, un anáglifo es una imagen de dos dimensiones capaz de provocar un efecto tridimensional, cuando se ve con lentes especiales.

En este diseño se puede mostrar a través de la salida HDMI, y dependiendo de los switches 7 y 6, la imagen captada por el sensor A (ojo izquierdo) dejando solo pasar el tono rojo, la imagen captada por el sensor B (ojo derecho) dejando solo pasar el tono azul y verde o el anáglifo:

- Si el switch 7 está a '1' y el 6 está a '0', se muestra la imagen captada por el sensor A únicamente dejando pasar el color rojo

Figura 45: Imagen del Ojo Izquierdo (sensor A), que solo deja pasar el Rojo

- Si el switch 7 está a '0' y el 6 está a '1', se muestra la imagen captada por el sensor B sin dejar pasar el color rojo, mostrando solo el color azul y verde.

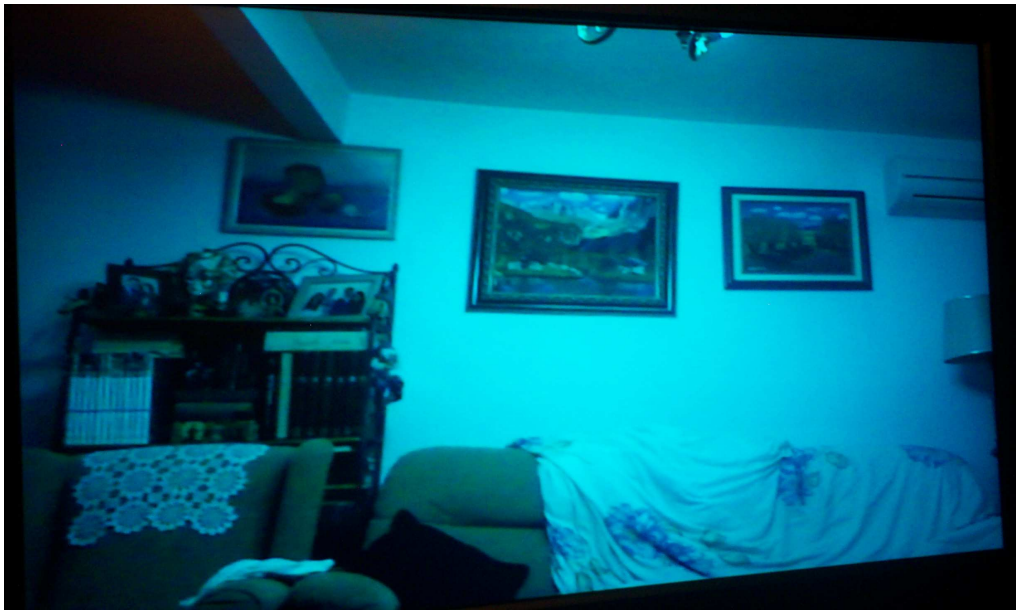


Figura 46: Imagen del Ojo Derecho (sensor B), que solo deja pasar el Azul y Verde

- Si ambos switches se encuentran a ‘1’ o a ‘0’, se muestra el anáglifo generado.

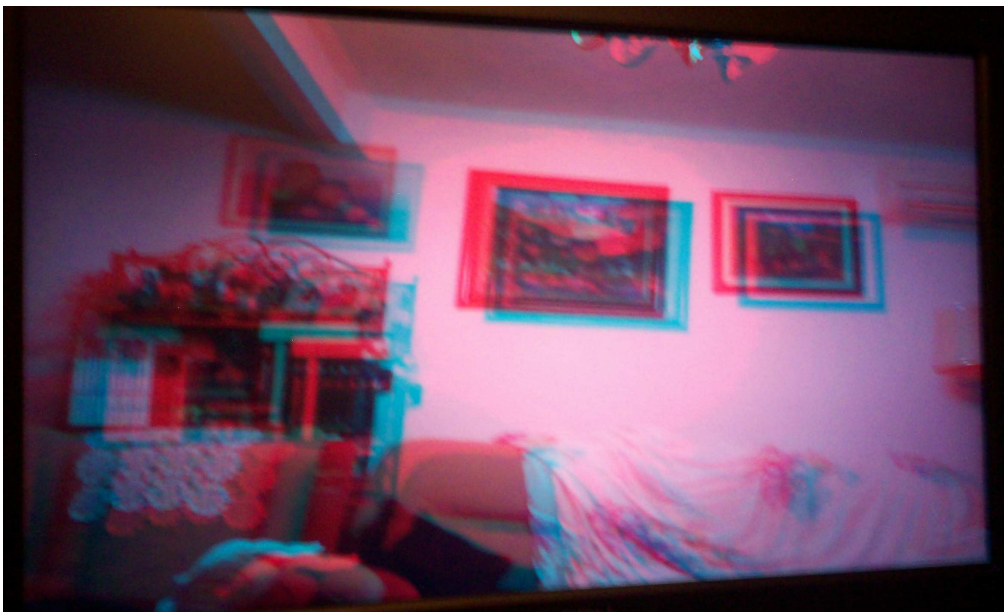


Figura 47: Anaglifo. Imagen del Ojo Izquierdo (sensor A) y del Ojo Derecho (sensor B) superpuestas

En la secuencia de imágenes anterior, se puede observar la primera imagen que simula el ojo izquierdo y solo deja pasar el color rojo; la segunda imagen que se ve, simula el ojo derecho, que solo deja pasar el color azul y verde; en la última imagen podemos ver el anáglifo que se genera superponiendo las dos imágenes anteriores; para poder apreciar la tridimensionalidad de forma adecuada, es necesario el uso de gafas anáglifo.

6. CONCLUSIONES Y TRABAJOS FUTUROS

6.1 CONCLUSIONES

El objetivo principal de este proyecto era el de estudiar un sistema de visión estéreo controlado por hardware; el objetivo era encontrar un sistema, que al contrario que los habituales sistemas de visión estéreo estuviera controlado por hardware y no por software, para así aumentar la velocidad de ejecución de los algoritmos de captura y procesamiento de imágenes. Además, como objetivos secundarios se intentarían desarrollar algunos diseños relacionados con la visión estéreo y que usaran nuestro sistema.

En este proyecto se ha conseguido el objetivo principal, ya que ese ha conseguido un sistema de visión estéreo controlado por hardware; el sistema compuesto por una FPGA, controladora del sistema, y una cámara de visión estéreo, como dispositivo de captación de imágenes, puede captar imágenes, procesarlas y mostrarlas por pantalla a través de un puerto de salida HDMI.

Para conseguir esto, se ha partido de un diseño básico de ejemplo, proporcionado por el fabricante de la FPGA. Inicialmente se estudió y comprendió cada uno de los módulos que conformaban dicho diseño, para comprender en que consistía y que resultados proporcionaba; entre los módulos principales se encontraban el modulo de captación de imágenes, CamCtl; el modulo encargado del almacenamiento en memoria de dichas imágenes, FBCtl, y el modulo encargado de transmitirá dichas imágenes al puerto de salida HDMI.

Como objetivos secundarios, uno de los uso que se pensaron para nuestro sistema, fue el Calcular mapas de Disparidad, con lo que se podrían calcular distancias en función de los niveles de gris de una imagen, pero no se llego a finalizar por completo, ya que tras muchas horas de estudio y de desarrollo, se concluyó que para realizar esto habría que rehacer por completo el diseño básico y comenzar el diseño desde cero, ya que el modulo de almacenamiento de las imágenes, no estaba preparado para este uso.

Otro de los usos que se pensó, fue un diseño que generara anáglifos, imágenes en dos dimensiones capaces de provocar un efecto tridimensional cuando se observan con unas gafas especiales. Este diseño sí se ha podido desarrollar por completo y los resultados se pueden ver en el capitulo anterior.

Leyendo las anteriores líneas, se puede decir que se ha cumplido tanto el objetivo principal como el secundario, ya que se ha conseguido un sistema de visión estéreo controlado por hardware capaz, además, de generar anáglifos en tiempo real.

6.2 TRABAJOS FUTUROS

El principal trabajo futuro que se puede realizar es desarrollar un diseño que calcule Mapas de Disparidad; usando este proyecto como referente y como “manual” del sistema de visión estéreo controlado por hardware, se puede realizar un proyecto que únicamente consista en realizar el diseño que calcula mapas de disparidad.

A partir de aquí, como trabajo futuro se puede desarrollar cualquier diseño que necesite de un sistema de visión estéreo y de una velocidad de ejecución superior a la que proporcionan los sistemas de visión estero controlados por software, como son los sistemas de reconocimiento facial o de detección y localización de obstáculos.

7. REFERENCIAS

[1] Digilent Inc,. AtlysTM Board Reference Manual. http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,836&Prod=ATLYS (Septiembre 2011)

[2] Digilent Inc,. VmodCAMTM Reference Manual. http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,648,931&Prod=VMOD-CAM (Septiembre 2011)

[3] Daniel Scharstein y Richard Szeliski. Stereo Visión Research. http://cat.middlebury.edu/stereo/. (Noviembre 2011)

[4] Estereoscopía. "Wikipedia enciclopedia libre". Disponible en la http://es.wikipedia.org/wiki/Estereoscop%C3%ADa. Noviembre 2011

[5] FPGA. "Wikipedia enciclopedia libre". Disponible en la http://es.wikipedia.org/wiki/Field_Programmable_Gate_Array. Octubre 2011

[6] VHDL. "Wikipedia enciclopedia libre". Disponible en la http://es.wikipedia.org/wiki/VHDL. Octubre 2011

[7] Manual Xilinx. Disponible en http://www.xilinx.com/itp/xilinx10/books/docs/qst/qst.pdf. Septiembre 2011

[8] Manual VHDL. http://www.dsi.fceia.unr.edu.ar/downloads/DDA/vhdl_PardoCarpio.pdf. Octubre 2011

[9] Ejemplo de Anaglifo. http://commons.wikimedia.org/wiki/File:Animacion_3b_anaglyphe.png?uselang=es. Agosto 2011

8. PRESUPUESTO

UNIVERSIDAD CARLOS III DE MADRID

Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor: Miguel Mariano García Zamora

2.- Departamento: Departamento de tecnología electrónica

3.- Descripción del Proyecto:

- Titulo: Estudio de un Sistema de Vision Estereo Controlador por Hardware

- Duración: 12 meses

Tasa de costes Indirectos: **20%**

4.- Presupuesto total del Proyecto (valores en Euros): 29904,3132Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F.	Categoría	Dedicación (hombres mes) [a)]	Coste hombre mes	Coste (Euro)	Firma de conformidad
García Zamora, Miguel Mariano	64216589-W	Ingeniero	9	2.694,39	24.249,51	

Hombres mes	**9**	**Total**	24.249,51

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)

Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Atlys™ Spartan-6 FPGA Development Board	269,45	100	12	60	53,89
VmodCAM - Stereo Camera Module	61,75	100	12	60	12,35
VHDCI Male-to-Male Cable	15,43	100	12	60	3,09
Xilinx ISE Design Suite 13.2	3.007,13	100	12	60	601,43
				Total	670,75

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado

B = periodo de depreciación (60 meses)

C = coste del equipo (sin IVA)

D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
-	-	-
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e)]

Descripción	Empresa	Costes imputable
-	-	-
	Total	0,00

[e)] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,…

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	24.250
Amortización	671
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	4.984
Total	29.904

9. ANEXOS

9.1 PARAMETROS DE IMAGEN DE LA CAMARA

Description	Variable Address		Bits	Default	
	Context A	Context B		Context A	Context B
Current Context (0 = A, 1 = B)		0xA702	7:0		0x0000
Output Width	0x2703	0x2707	15:0	800	1600
Output Height	0x2705	0x2709	15:0	600	1200
Sensor Row Start	0x270D	0x272F	15:0	0	4
Sensor Row End	0x2711	0x2733	15:0	1213	1211
Sensor Column Start	0x270F	0x2731	15:0	0	4
Sensor Column End	0x2713	0x2735	15:0	1613	1611
Read Mode	0x2719	0x273B	15:0	0x046C	0x0024
x-bin enable			11	0x0	0x0
xy-bin enable			10	0x1	0x0
x odd increment			7:5	0x3	0x1
y odd increment			4:2	0x3	0x1
vertical flip (0=normal)			1	0x0	0x0
horizontal flip (0=normal)			0	0x0	0x0
Crop X0	0x2751	0x275F	15:0	0	0
Crop X1	0x2753	0x2761	15:0	800	1600
Crop Y0	0x2755	0x2763	15:0	0	0
Crop Y1	0x2757	0x2765	15:0	600	1200
Output format	0x2795	0x2797	15:0	0x0000	0x0000
processed Bayer mode			8	0x0	0x0
RGB output format 0x0=RGB565 0x1=RGB555 0x2=RGB444x 0x3=RGBx444			7:6	0x0	0x0
RGB/YUV 0x1=RGB 0x0 = YUV			5	0x0	0x0
Use CCIR656 codes when bypassing FIFO			4	0x0	0x0
Monochrome output			3	0x0	0x0
Progressive Bayer			2	0x0	0x0
Swap chrominance/luminance bytes in YUV, swap odd/even bytes in RGB			1	0x0	0x0
Swap Cr/Cb in YUV, swap R/B in RGB			0	0x0	0x0